University of Alberta

ANALOG FFT INTERFACE FOR ULTRA-LOW POWER ANALOG RECEIVER ARCHITECTURES

by

Nima Sadeghi

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**

Department of Electrical and Computer Engineering

Edmonton, Alberta
Spring 2007

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-30013-8
Our file Notre référence
ISBN: 978-0-494-30013-8

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

To whom I trust in,
The one who's my everything,
and
Those who are seeking the truth and the beauty of life.

Abstract

After the emergence of iterative decoding in the late 1990's, analog decoding design became a powerful alternative to conventional digital implementation. Our goal is to design and implement an entire analog receiver including an analog decoder and a low power analog Fast Fourier Transform (FFT) input interface. The analog decoder part of the design was recently demonstrated by a doctoral candidate. This project focuses on the design of a novel analog FFT processor. The methodology of the design is based on mutually considering system and circuit levels. We simulated the system considering different circuit issues. We modeled an accurate mathematical input referred mismatch source for the N-FFT. We showed that the higher radix FFT structures like DFT have reduced sensitivity to mismatch and reduced number of current mirrors. Subsequent to our work a 180-nm realization was designed and fabricated in collaboration with another M.Sc. candidate.

Acknowledgements

Firstly, I would like to thank Drs. Christian Schlegel and Vincent C. Gaudet for their supervision, guidance, and funding for the project presented in this thesis. I would also like to thank all the colleagues working at the High Capacity Digital Communications Laboratory for their kind support. In addition, I would like to especially thank my previous professors in Sharif university, Drs. Masoud Jahanbegloo, Mehrdad Sharif-Bakhtiar, and Mohamad H. Alavi, who taught me the principles of circuit design.

Finally I would like to thank my family living in different parts of the world, Iran, Germany, and U.S.A., for encouraging me on my way and helping me to enjoy my life here in Canada. I would also like to thank my special friends, all over the world, for their being there, and here.

Table of Contents

1 Introduction 1

2 Background and Motivation 3
- 2.1 Analog Receiver Design and Implementation . . . 3
  - 2.1.1 System On A Chip . . . 3
  - 2.1.2 Energy Scavenging Approach . . . 4
- 2.2 Analog Decoder (Receiver Core) . . . 5
  - 2.2.1 Novel Recent Codes . . . 5
  - 2.2.2 Digital Decoder Implementation . . . 5
  - 2.2.3 Analog Decoder Implementation . . . 6
- 2.3 Low-power Analog Interface: Analog FFT . . . 7
  - 2.3.1 OFDM Advantages and Analog FFT Processor . . . 7
- 2.4 Design Methodology . . . 8
  - 2.4.1 System Level Transceiver Model . . . 9
  - 2.4.2 Circuit Level Modeling . . . 14
- 2.5 Chapter Conclusion . . . 17

3 Analog FFT Processor 19
- 3.1 Radix-2 FFT Butterfly Structure . . . 19
  - 3.1.1 8-FFT Structure . . . 19
  - 3.1.2 Extending Algorithm for Larger FFT . . . 23
  - 3.1.3 Index Sorting Algorithm . . . 23
  - 3.1.4 256-FFT Structure . . . 25
- 3.2 Expandable Analog FFT Circuit Design . . . 28
  - 3.2.1 Basic Circuit Block for Mirrors and Weighting Factors . . . 29
  - 3.2.2 NMOS/ PMOS Consecutive Stages . . . 30
  - 3.2.3 Dummy Mirrors for Each WF Block . . . 30
  - 3.2.4 Routing Algorithm for the Entire 256-FFT Circuit . . . 31
- 3.3 Current Mirror Model . . . 34
  - 3.3.1 Current Scaling at Every FFT Stage . . . 37
- 3.4 Mismatch model . . . 38
  - 3.4.1 Mathematical View for Mismatch Model . . . 39
  - 3.4.2 Input Referred Mismatch Model . . . 43
- 3.5 Power Consumption . . . 48
- 3.6 Chapter Conclusion . . . 49

4 Simulations and System Performance 50
- 4.1 Differential BPSK System Performance . . . 51
- 4.2 BER/SNR FFT Simulations . . . 53
  - 4.2.1 Current Mirror Model for 256-FFT Simulations . . . 53
  - 4.2.2 Current Scaling in every FFT Stage for 256-FFT Simulations 54
  - 4.2.3 Simplification of WFs for 8-FFT up to 256-FFT Simulations 56

4.2.4 Mismatch Model for 8-FFT up to 256-FFT Simulations . . . 57
4.2.5 Radix-4 and Radix-16 Estimated Mismatch Model Simulations for 256-FFT . . . 60
4.2.6 Decoder Simulations . . . 62
4.3 System Performance Matrix . . . 63
4.4 Chapter Conclusion . . . 65

5 Conclusions **66**
5.1 Contributions . . . 66
5.2 Future Work . . . 68

Bibliography **70**

List of Tables

2.1 Energy efficiency and size efficiency of recent analog/digital decoders. . . 4

3.1 Number of transistors used in each FFT stage. 31
3.2 NMOS mirror characteristic over the wide input/output range. 36
3.3 PMOS mirror characteristic over the wide input/output range. 36
3.4 Input referred mismatch variance for N-bit (radix-2) FFT from 2-FFT up to 256-FFT. 45
3.5 Estimation of Input referred mismatch variance for different radix N-FFT structures. 46
3.6 Input referred mismatch variance comparison of N-FFT for radix-2, 4, and 16. 47
3.7 The comparison of the Input referred mismatch variances for different types of the 256-FFT. 47
3.8 The comparison of the number of current mirrors in 256-FFT for different radix structures. 48

4.1 The system performance matrix for having BER loss of 0.5 dB for different values of V_{th} mismatch. Con_{str} is $\frac{2V_{th}}{V_{gs}-V_{th}}$, F_{imp} is the standard deviation improving factor for different FFT types, P_{factor} is a factor representing an extra power consumption due to increasing V_{dd} from 1.8 (V), and S_{factor} is a factor representing an extra Silicon area due to increasing L from minimum feature size, $0.18\mu m$. 64

List of Figures

2.1 Comparison of energy efficiency versus size efficiency of recent Analog/ Digital Decoders. 6
2.2 OFDM communication system model. 7
2.3 Analog iterative decoding compared to digital iterative decoding 14
2.4 Basic block of analog differential multiplier with translinear loop [1]. . . . 16

3.1 Butterfly Diagram of an 8-FFT. The inputs and outputs are complex differential values and $W_8{}^1$, $W_8{}^2$ and $W_8{}^3$ are complex constants [2]. 20
3.2 8-FFT Weighting Factors on trigonometric circle. 21
3.3 Modified Butterfly Diagram for 8-FFT 21
3.4 Butterfly Structure of FFT. 22
3.5 Index sorting algorithm for 8-FFT. 23
3.6 Binary representations of 8-FFT's input and output, and their bit reversal dependency. 24
3.7 Generating Weighting Factors for different butterfly stages in the 256-FFT. 25
3.8 The circular view of the Weighting Factors on the unity circle. 27
3.9 Current mirror: basic circuit to implement FFT. 28
3.10 First basic block of the upper half part of the 8-FFT stage. 33
3.11 First basic block of the lower half part of the 8-FFT stage. 33
3.12 Basic model to find the mirror characteristic over the large input range. . . 35
3.13 I_{out} versus I_{in} characteristic of NMOS current mirror with PMOS load. . 37
3.14 Input Referred Mismatch model for the N-FFT block 38
3.15 Log-normal distribution for weak inversion operation, 10% Mismatch. Its mean is a positive value 0.24 and its variance is about double 41
3.16 Log-normal distribution for weak inversion operation, 5% Mismatch. Log-normal distribution has small difference in mean and variance compared to normal distribution. 42
3.17 Log-normal distribution for weak inversion operation, 1% Mismatch. Log-normal distribution can be estimated by normal distribution. 42
3.18 Mismatch for the 2-FFT block. Each current mirror has its ε_{sci} where s indicates the stage, c is either 1 or 2 representing the first or second copy of an input i. 43
3.19 Mismatch for the 4-FFT block. Each current mirror has its ε_{sci} where s indicates the stage, c is either 1 or 2 representing the first or second copy of an input i. 43
3.20 Input Referred Mismatch Variance for the N-FFT block 45

4.1 Ideal 256-FFT performance; Coherent demodulation compare to differential demodulation. 51
4.2 Ideal 256-FFT design for coherent versus differential design. 52
4.3 Statistical simulation of 256-FFT having current mirror model for different bias currents. 53

4.4 Statistical simulation of 256-FFT with scaling/non-scaling current at each stage for 100 (pA) bias current. The dashed line is a non-scaled version and the solid line is a scaled version. 54
4.5 Statistical simulation of 256-FFT with scaling/non-scaling current at each stage for 10 (nA) bias current. The dashed line is a non-scaled version and the solid line is a scaled version. 55
4.6 Statistical simulation of 256-FFT with scaling/non-scaling current at each stage for 100 (nA) bias current. The dashed line is a non-scaled version and the solid line is a scaled version. 55
4.7 Statistical simulation of FFT, from 8 to 256 bit, with only three different values for WFs, 0.4, 07, and 0.9. 56
4.8 Mismatch simulation of 8-FFT for different threshold voltage variations, 10%, 2.2%, and 1%. 58
4.9 Mismatch simulation of FFT, from 8 to 64-FFT with 1% mismatch. 59
4.10 Mismatch simulation of FFT, from 64 to 256-FFT with 0.1% mismatch. . 59
4.11 Mismatch simulation of 8-FFT in weak inversion for different threshold voltage variations, 0.22%, 0.17%, and 0.1%. 60
4.12 Mismatch simulation of radix-2 256-FFT for 0.1%, 0.2%, and 1%, mismatch. 61
4.13 Mismatch simulation of 256-FFT for radix-4, radix-16 and DFT structure. 61
4.14 Radix-2 256-FFT performance using TPC Decoder for BPSK dimodulation 63

List of Abbreviations

Abbreviation	Definition
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
CMOS	Complementary metaloxidesemiconductor
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
IFFT	Inverse Fast Fourier Transform
KCL	Kirchhoff's Current Law
LDPC	Low Density Parity Check
LLR	Log Likelihood Ratio
OFDM	Orthogonal Frequency Division Multiplexing
PDF	Probability Density Function
QPSK	Quadrature Phase Shift Keying
SNR	Signal to Noise Ratio
TPC	Turbo Product Code
WF	Weighting Factor

List of Symbols

Symbol	Definition
$S(t)$	Complex equivalent baseband signal
$S_I(t)$	In-phase baseband signal
$S_Q(t)$	Quadrature baseband signal
T_s	Sampling Time
d_n	Complex data symbol
a_n	Real data symbol
b_n	Imaginary data symbol
f_k	Frequency channels
Δf	Frequency spacing
c_k	Coded information bit
θ	Phase offset
Φ_k	Phase reference
y_k	Received noisy signal
r_k	Recovered data
I_D	Drain current
V_{gs}	Gate Source Voltage
V_{ds}	Drain Source Voltage
V_{th}	Threshold Voltage
x_k	Complex Differential input signal
${W_N}^k$	FFT Complex multiplicand
WF_{ki}	FFT In-phase multiplicand
WF_{kq}	FFT Quadrature multiplicand
I_{ki+}	In-phase Plus current
I_{ki-}	In-phase Minus current
I_{kq+}	Quadrature Plus current
I_{kq-}	Quadrature Minus current
ε	Normal distributed random variable of mismatch
δ_ε^2	ε Variance
δ_N^2	N-FFT Input referred mismatch Variance
Con_{str}	Normalizing factor in Strong inversion
Con_{sub}	Normalizing factor in Subthreshold
V_T	Temperature coefficient
P_b	Probability of error

Chapter 1

Introduction

Due to the low power consumption constraint of recent System On a Chip (SOC) designs, analog implementations of communication receivers recently have been considered and compared to their digital counterparts. This research has been focused on building a low power analog input interface for an analog receiver which takes received signals and generates the Log-Likelihood Ratios (LLR) required by an analog decoder.

In Chapter 2 we provide a brief background introduction to the idea of analog receiver design. As a motivation we compare conventional digital implementation of decoders, the main part of the receiver, with recent analog implementations of decoders. We introduce some novel codes which are suitable for analog implementations and we discuss the potential possibility of having a receiver with such a low power consumption suitable for energy scavenging design. Also in this chapter we explain our design methodology, which is divided into system and circuit level design. At the system level we propose our communication system model which uses Orthogonal Frequency Division Multiplexing (OFDM) transmission with differential Binary Phase Shift Keying (BPSK) modulation. Having OFDM as a transmission format requires an FFT at the receiver side. At the circuit level we explain how an analog FFT can be implemented using simple current mirrors as basic blocks.

Chapter 3 describes the novel design of an analog FFT processor. In the first section we explain the butterfly structure of an 8-symbol FFT (denoted 8-FFT) and we provide the extending algorithm for larger FFTs. Then we consider the 256-FFT structure as a system model for our analog FFT processor. The next section is about

basic circuit blocks for expandable analog FFTs, which are current mirrors. Then we discuss some circuit issues about where to use NMOS or PMOS transistors and a routing algorithm for an entire N-FFT. In the next section of chapter 2 we derive a model of the current mirrors which provides an output function of input signals to use this function on our system code, written in Matlab, to examine the behavior of our actual circuits. The next section talks about threshold voltage mismatch as a dominant source of mismatch in a transistor pair. We consider it as a normally distributed random variable with zero mean and a specific variance. We also provide an input referred mismatch model for our analog N-FFT. At the end of this chapter we discuss the power consumption of the proposed analog FFT circuit.

In Chapter 4 we discuss our simulation results and the system performance under the different design issues. First we explain the performance of differential BPSK and compare it to a Quadrature Phase Shift Keying (QPSK) modulation system and discuss the tradeoffs between their Bit Error Rate (BER) performance and their spectral efficiency. Then we look at FFT simulations under different circuit issues. We examine the BER performance of FFT to see how sensitive it is to the different bias currents, the current mirror model, and the simplification of Weighting Factors (WF) inside the FFT structure. We compare the impact of mismatch on the FFT performance in strong and weak inversion. We also show that the BER curve of our input referred mismatch model follows the curve of the FFT, which has mismatch at each current mirror. We discuss using a higher radix FFT structure to minimize the effect of mismatch. At the end we show how the decoder improves the performance of FFT due to mismatch loss.

Chapter 5 concludes the thesis and provides future research directions.

In this work first we look at the system level design and its mathematical representation. Then we explain the FFT structure and some circuit considerations. At the end we discuss the system performance and the simulation results.

Chapter 2

Background and Motivation

2.1 Analog Receiver Design and Implementation

Due to their capacity-approaching potential, latest-generation error control codes such as Turbo Codes [3] and LDPC Codes [4] have been incorporated into many recent data communications standards such as IEEE 802.11a, IEEE 802.16 and DVB-S2, and have been proposed for the emerging IEEE 802.3 10GBASE-T Ethernet standard [5].

Conventional VLSI implementations of iterative decoders have largely used digital circuitry due to the ease of design of such circuits [6]. However, these implementations consume large amounts of power and silicon area [1]. Because of these limitations, analog decoding circuits have recently been considered as an alternative to digital processing. Initial results for analog decoders by a number of research groups have shown impressive results. Some decoders consuming approximately one orders of magnitude less energy per transmitted bit than comparable digital decoders, at similar error rate performances [1, 7, 8, 9].

2.1.1 System On A Chip

The goal of the proposed project is to move towards a full system-level integration of analog decoders with other basic communications receiver components, while maintaining the power consumption advantages of analog decoders. This will involve the incorporation and interfacing of a receiver front-end with an analog decoder. Such a task will require research innovations both at the systems level,

Error Control Code Type	CMOS Technology	Energy Efficiency (nJ/b)	Normalized Size (mm^2/N)
$(8,4)$ Hamming Trellis Graph[7]	$0.18\mu m$ Analog	0.041 Core	$2.5(10^{-4})$ Core
$(8,4)$ Hamming Factor Graph [7]	$0.18\mu m$ Analog	0.218 Core	$2.5(10^{-3})$ Core
$(16,11)$ Hamming [1]	$0.18\mu m$ Analog	0.02 Core	$1.5(10^{-2})$ Core
$(8,4)$ Hamming Tail-biting [9]	$0.5\mu m$ Analog	1 Core	$1.012(10^{-1})$ Core
$(16,11)^2$ Turbo Product Code [1]	$0.18\mu m$ Analog	0.17 Core	$1.56(10^{-2})$ Core
$(32,8)$ LDPC [8]	$0.18\mu m$ Analog	0.83 Chip	$1.78(10^{-2})$ Core
Turbo Code (40-Bit, N=132) [11]	$0.35\mu m$ Analog	3.4 Core	$3.11(10^{-2})$ Core
Turbo Code (16-Bit, N=48) [12]	$0.35\mu m$ Analog	13.9 Chip	$2.75(10^{-2})$ Core
$(1024,512)$ LDPC [6]	$0.16\mu m$ Digital	1.38 Core	$5.13(10^{-2})$ Core
$(2048,1024:1792)$ Programmable LDPC [13]	$0.18\mu m$ Digital	2.46	$7(10^{-3})$ Core

Table 2.1: Energy efficiency and size efficiency of recent analog/digital decoders.

where the choice of receiver structure will impact on overall system performance and power consumption, and at the circuits level, since some receiver blocks will require novel computational processing nodes.

2.1.2 Energy Scavenging Approach

The target application behind this project is an ultra low-power radio receiver and data decoder which could operate with power levels so low that energy scavenging methods [10] could provide sufficient power to operate the entire receiver. Such a receiver might well find application in sensor networks or medical monitoring, where extremely low power consuming communications devices will be a necessity.

In Table 2.1 we compare the major recent analog and digital decoders in terms of energy efficiency versus size efficiency. The measure for power is energy efficiency (nJ/bit), amount of nano-Jules per decoded information bit, and the measure for

area is the size of the core normalized with the code length, (mm^2/N unit code length).

More explanations about recent analog decoders and their characteristics, advantages of analog design compared to their digital counterparts, topology of ultra low power circuits using subthreshold CMOS techniques and important considerations of analog FFT design will be covered in the following sections.

2.2 Analog Decoder (Receiver Core)

2.2.1 Novel Recent Codes

From the time when Shannon introduced his channel capacity as a limit for any communication channel in 1948 [14], people have been trying to produce a system which achieves this bound. By introducing Turbo codes, in 1993, Berrou *et. al* first presented practical capacity-approaching codes for the additive white Gaussian noise (AWGN) channel [3, 15]. After that other types of known codes such as Block Product codes and LDPC codes were quickly realized to also approach capacity on the AWGN channel using iterative decoding [16, 17, 18].

In addition to approaching capacity, the complexity and cost of a coding system must be considered to define the efficiency of a system [19].

2.2.2 Digital Decoder Implementation

With digital implementation these iterative algorithms require digital circuitry performing hundreds or thousands of calculations per decoded bit, i.e. for desired large data rates we need a very high clock frequency decoder. However such a decoder has its problems such as dissipating a lot of power, generating large amounts of heat and probably causing high-frequency interface in other adjacent circuits [1, 6]. For example, a high speed Analog-to-Digital Converter (ADC) is required for a digital receiver, which is a power hungry component for digital implementation.

2.2.3 Analog Decoder Implementation

An alternative solution to have a high-speed decoder is a fully parallel analog decoder. A parallel implementation of a decoder is desirable due to intrinsic parallel and distributed nature of iterative decoding algorithms. Therefore, by parallelizing, using analog circuitries, we can reduce the operating time, which can be compared to the operating clock of the digital decoders, by an amount proportional to data's block rate [1].

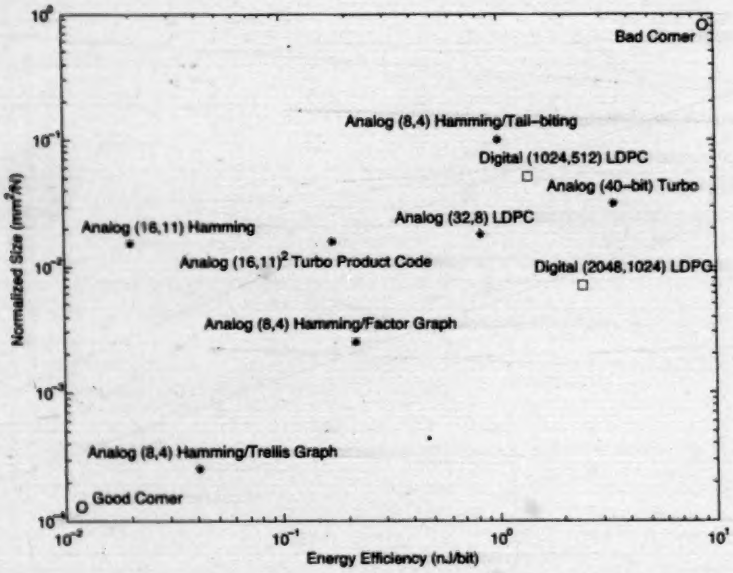

Figure 2.1: Comparison of energy efficiency versus size efficiency of recent Analog/ Digital Decoders.

Fig. 2.1 illustrates a graphical performance comparison in terms of energy efficiency versus size efficiency of the major recent analog/digital decoders based on the data given in Table 2.1. We represent energy efficiency (nJ/bit) on the x-axis, and normalized size, (mm^2/N) on the y-axis. Most of the analog decoders are closer to the good corner on this graph compared to their digital counterparts.

For ultra-low power applications Winstead *et. al* have designed an analog decoder with the length of 256, $(16, 11)^2$ Turbo Product Code (TPC) [1], and we want to build an analog 256-FFT interface to extend this project eventually to obtain an analog receiver.

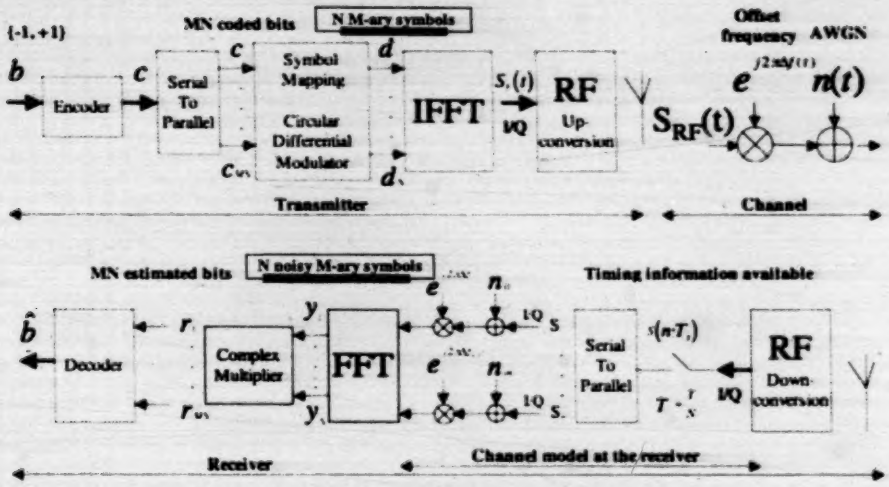

Figure 2.2: OFDM communication system model.

2.3 Low-power Analog Interface: Analog FFT

An adequate high-speed analog interface needs to be built which takes a received noisy signal and generates the log-likelihood ratios required by an analog decoder. This interface needs to observe strict low power requirements and must not adversely affect the performance of the analog decoder.

2.3.1 OFDM Advantages and Analog FFT Processor

We propose to use OFDM as the transmission format for a wireless test chip [20]. This requires that the receiver processes received signal samples by a Discrete Fourier Transform (DFT) to generate the LLR values required by the analog decoder. The DFT can be represented by a graph that has similarities to the graph describing the error control code. These graphs describe the high level layout of the analog decoder circuits and it is expected that the DFT design can be implemented using comparable analog circuits as those used in analog decoders. We propose to use an FFT, Fast Fourier Transform version of the DFT because FFT's computational complexity is of the order *NlogN* which is remarkably lower than the DFT's, $O(N^2)$, for a large number of bits [21, 22]. The analog implementation of an

N-FFT has some advantages compared to its digital implementation; The addition operation in digital design is a computation while in analog design the addition is tying wires together hence it is cost free, also the number of wires per input in analog design is two for having differential values while in digital design it is equal to $log_2(Q)$, where Q is the number of quantization levels, which leads us to the higher number of wires per input. The design and implementation of an analog FFT processor is the novel part of this research since all the existing FFT processors have been implemented using digital circuits.

There are several ways to choose a transmission format for our receiver. We propose to use Orthogonal Frequency Division Multiplexing known as OFDM. OFDM is a version of frequency division multiplex (FDM) multicarrier modulation (MCM). This modulation, by using a large number of parallel narrow-band orthogonal sub-carriers, improves the performance of transmission such that no intercarrier guard bands are needed. It provides controlled overlapping of bands and maximum spectral efficiency. Also it has been adopted for various standards such as DSL, 802.11a, DAB, DVB. Some advantages of using OFDM are an easy implementation using IFFTs (Inverse Fast Fourier Transforms), efficiency in dealing with multi-path propagation and robustness against narrow-band interference [20]. According to these features, OFDM is a proper candidate for the transmission format of information signals used in our receiver.

The system level is the first level of design and after these considerations, we can look at the signal specification of the system.

2.4 Design Methodology

To be able to complete our entire analog receiver design, a well thought-out methodology is unavoidable. It is divided into three separate levels that are system level, signaling level and circuit level in each of which we focus on different levels of the design with specific considerations. At the system level we focus on theoretical methods that can achieve our goal such as choosing a proper modulation scheme with specific constellation like QAM, BPSK or MSK, using a well-behaved er-

ror correcting code such as Turbo codes or LDPC codes, and choosing a best-fit transmission format out of existing scheme like OFDM, CDMA. Representing our existing model at the system level by electrical signals, all input, output and noise signals are considered as currents or voltages, leading us to the signal level of design to implement our design using CMOS technology. At the circuit level we look at circuit topology, the way we can implement our design using physical components, and their non-ideal behavior. We explain these design levels in the following chapters.

2.4.1 System Level Transceiver Model

At the system level we look at the entire communication system model which consists of a transmitter, communication channel, and receiver shown in Fig. 2.2. We explain each individual part in this model and compute the bit error rate performance results.

At the transmitter binary information bits are encoded by an error correcting code such as a Turbo or LDPC code. A serial-to-parallel data converter gives *M*N* coded bits to a symbol mapping block to generate N 2^{M-ary} symbols. We will concentrate on BPSK, i.e., *M=1*. The mapping is differential to avoid phase recovery. The inverse fast Fourier transform (IFFT) creates an OFDM transmission signal. The IFFT creates both in-phase and quadrature channels. After RF up-conversion, the complex equivalent baseband signal $S(t)$ with a symbol period [0,T], containing $S_I(t)$ and $S_Q(t)$, is transmitted.

At the receiver, the RF signal is down-converted. The signal is sampled at times nT_s producing $S(nT_s) = S_n$ where $T_s = \frac{T}{N}$. We assume that the timing information is available at this point. After sampling, we can model the channel at the receiver. We add AWGN, and consider frequency offsets, $e^{j2\pi\Delta f_k}$, multiplying each sample. The n noisy samples are demodulated by an N point FFT processor. For symbol detection, we use a circular differential demodulator in which we multiply adjacent samples to cancel out the unknown common phase offset. The estimated samples are delivered to the error control decoder to recover the original transmitted bits.

Mathematical View of OFDM Transmitter

The data symbols d_n for the different frequency channels are in general complex, i.e., $d_n = a_n + jb_n$. Using the IFFT generates the I/Q baseband waveform $S_b(t) = S_I(t) + jS_Q(t)$ where

$$S_b(t) = \left(\sum_{k=0}^{N-1} d_k e^{j\omega_k t}\right); \quad 0 \le t \le T \tag{2.1}$$

$$S_I(t) = \mathrm{Re}\left(\sum_{k=0}^{N-1} \mathrm{d_k e^{j\omega_k t}}\right) = \sum_{k=0}^{N-1} \left(\mathrm{a_k \cos(\omega_k t) - b_k \sin(\omega_k t)}\right); \quad 0 \le \mathrm{t} \le \mathrm{T} \tag{2.2}$$

$$S_Q(t) = \mathrm{Im}\left(\sum_{k=0}^{N-1} \mathrm{d_k e^{j\omega_k t}}\right) = \sum_{k=0}^{N-1} \left(\mathrm{a_k \sin(\omega_k t) + b_k \cos(\omega_k t)}\right); \quad 0 \le \mathrm{t} \le \mathrm{T}. \tag{2.3}$$

In (2.1) $\omega_k = 2\pi f_k$, where $f_k = k\Delta f$, and $\Delta f = \frac{1}{T}$ is the frequency spacing to generate the different baseband frequency channels.

After generating baseband OFDM signals, we up-convert them by the RF carrier frequency f_c,

$$S_{RF}(t) = \mathrm{Re}\left(\sum_{k=0}^{N-1} S_b(t) e^{j\omega_c t}\right) = \sum_{k=0}^{N-1} \left(S_\mathrm{I}(t)\cos(\omega_c t) - S_Q(t)\sin(\omega_c t)\right) \; 0 \le t \le T. \tag{2.4}$$

In (2.4) $\omega_c = 2\pi f_c$. We transmit

$$S_{\mathrm{RF}}(t) = \mathrm{Re}\left(\sum_{k=0}^{N-1} S_b(t) e^{j\omega_c t}\right) \mathrm{rect(t/T)}, \tag{2.5}$$

where $\mathrm{rect}(t/T)$ is 1 if $0 \le t \le T$ and zero elsewhere.

To discuss the receiver architecture and the recovery technique we look at the baseband received OFDM signals after down-conversion in which f_c is removed. We consider it as a complex envelope signal, $S_b(t)$.

Mathematical View of the Receiver

At the receiver we sample the signal $S_b(t)$ at $t_n = nT_s$, where $T_s = \frac{T}{N}$ to create the discrete samples for the FFT given by:

$$S_n = \sum_{k=0}^{N-1} d_k e^{j2\pi k \Delta f n T_s}, 0 \leq n \leq N-1. \quad (2.6)$$

We rewrite (2.6) by substituting $d_n = a_n + jb_n$, $\Delta f = \frac{1}{T}$ and $T_s = \frac{T}{N}$:

$$S_n = \sum_{k=0}^{N-1} (a_k + jb_k) e^{\frac{j2\pi kn}{N}}, 0 \leq n \leq N-1. \quad (2.7)$$

Both the real and imaginary parts of S_n, in general, contain our transmitted data, a_k and b_k:

$$S_{In} = \sum_{k=0}^{N-1} \left(a_k \cos\left(\frac{2\pi kn}{N}\right) - b_k \sin\left(\frac{2\pi kn}{N}\right)\right); \quad 0 \leq n \leq N-1 \quad (2.8)$$

$$S_{Qn} = \sum_{k=0}^{N-1} \left(a_k \sin\left(\frac{2\pi kn}{N}\right) + b_k \cos\left(\frac{2\pi kn}{N}\right)\right); \quad 0 \leq n \leq N-1. \quad (2.9)$$

We can recover our transmitted data using the FFT as follows:

$$y_k = \hat{d}_k = \frac{1}{N} \sum_{n=0}^{N-1} S_n e^{\frac{-j2\pi kn}{N}}. \quad (2.10)$$

Substituting S_n from (2.7) we obtain

$$y_k = \hat{d}_k = \frac{1}{N} \sum_{n=0}^{N-1} \sum_{l=0}^{N-1} d_l e^{\frac{j2\pi n(l-k)}{N}}. \quad (2.11)$$

By reordering the sums in (2.11) we obtain

$$y_k = \hat{d}_k = \frac{1}{N} \sum_{l=0}^{N-1} d_l \sum_{n=0}^{N-1} e^{\frac{j2\pi n(l-k)}{N}}; \quad (2.12)$$

and

$$y_k = \hat{d}_k = \frac{1}{N} \sum_{l=0}^{N-1} d_l N \delta(l-k) = d_k. \quad (2.13)$$

Thus far we have assumed that the channel is ideal and we showed that data recovery can be accomplished via an FFT transformation.

Phase Offset Consideration

Now consider the receiver samples when there is a phase offset θ. The new samples S_n are now given from (2.7) by:

$$S_n = \sum_{k=0}^{N-1} d_k e^{\frac{j2\pi kn}{N}} e^{j\theta}; \quad 0 \le n \le N-1 \tag{2.14}$$

where we expand S_n into:

$$\begin{aligned} S_n = &\sum_{k=0}^{N-1} \left(a_k \cos\left(\frac{2\pi kn}{N} + \theta\right) - b_k \sin\left(\frac{2\pi kn}{N} + \theta\right)\right) \\ &+ j \sum_{k=0}^{N-1} \left(a_k \sin\left(\frac{2\pi kn}{N} + \theta\right) + b_k \cos\left(\frac{2\pi kn}{N} + \theta\right)\right), \end{aligned} \tag{2.15}$$

and

$$\begin{aligned} S_n = &\sum_{k=0}^{N-1} \Big(a_k \cos(\theta)\cos(\frac{2\pi kn}{N}) - a_k \sin(\theta)\sin(\frac{2\pi kn}{N}) - \\ &- b_k \sin(\theta)\cos(\frac{2\pi kn}{N}) - b_k \cos(\theta)\sin(\frac{2\pi kn}{N})\Big) \\ &+ j \sum_{k=0}^{N-1} \Big(a_k \sin(\theta)\cos(\frac{2\pi kn}{N}) + a_k \cos(\theta)\sin(\frac{2\pi kn}{N}) + \\ &+ b_k \cos(\theta)\cos(\frac{2\pi kn}{N}) - b_k \sin(\theta)\sin(\frac{2\pi kn}{N})\Big), \end{aligned} \tag{2.16}$$

which is a rotation of the original data, a_k and b_k, by θ. Here we assume that the phase offset is constant for all n samples. If this phase offset is not known we may use differential modulation.

Consider the following differential QPSK modulation as example, in which the data are modulated differentially as

$$d_k = e^{j\Phi_k} = e^{j(\Phi_{k-1} + \Delta\Phi_k)}; \quad \Delta\Phi_k = 0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}, \tag{2.17}$$

where Φ_{k-1} is a phase reference for symbol d_k, and $\Delta\Phi_k$ is our original coded information bit, c_k. If we conjugate each coming complex symbol at the receiver and multiply it by the next symbol, we obtain the original data regardless the phase offset:

$$r_k = (y_{k-1}^*)(y_k) = (d_{k-1}^*)e^{-j\theta}d_k e^{j\theta} = e^{-j(\Phi_{k-1})}e^{j(\Phi_{k-1}+\Delta\Phi_k)} = e^{j(\Delta\Phi_k)} = c_k. \tag{2.18}$$

To generate the phase reference for the first symbol d_1 we use a tail-biting method; we consider c_N as a refrence for d_1. At the circular differential modulator we add the phase of adjacent coded information bits c_k, using multiplication

$$d_k = c_{k-1} \cdot c_k; \quad 2 \leq k \leq N \tag{2.19}$$

$$d_1 = c_N \cdot c_1; \quad k = 1, \tag{2.20}$$

and at the demodulator we subtract the phase of adjacent noisy symbols y_k, using complex multiplication

$$y_{k-1}^* \cdot y_k = c_k; \quad 2 \leq k \leq N \tag{2.21}$$

$$y_N^* \cdot y_1 = c_1; \quad k = 1. \tag{2.22}$$

After differential demodulation, N estimated symbols are delivered to the decoder to extract information bits as depicted in Fig.2.2.

The phase offset is taken care of by using this differential scheme at the cost of N=256 number of complex multipliers at the receiver front end between the FFT processor and the decoder.

Then we model our analog FFT design using Matlab, considering how different parameters affect its result due to non ideal characteristics of the circuit, mismatch for instance, as a specific function. As we will mention later on, the basic block of the analog FFT processor is its current mirror and we try to find the piece-wise linear function of our current mirror for the different input values and model this new function in our Matlab code to consider the effect of non ideal characteristics. The purpose is to design our analog FFT block in such a way as to be tolerant to device non idealities in terms of BER performance.

2.4.2 Circuit Level Modeling

The strength of analog computation for decoder implementation is related to the fact that it is possible to generate the basic operations needed for iterative decoding by using very simple analog circuits. The two fundamental operations are multiplication and addition in iterative decoding using the sum-product algorithm. By using a simple Gilbert Multiplier circuit based on the translinear concept, which is explained in more detail later in this section, we implement multiplication operations. Addition can be done by tying two wires based on Kirchhoff's Current Law (KCL) to sum these currents [23]. Thus the implementations of analog iterative decoders are smaller than their traditional digital counterparts, and often have less power consumption by about one order of magnitude [1].

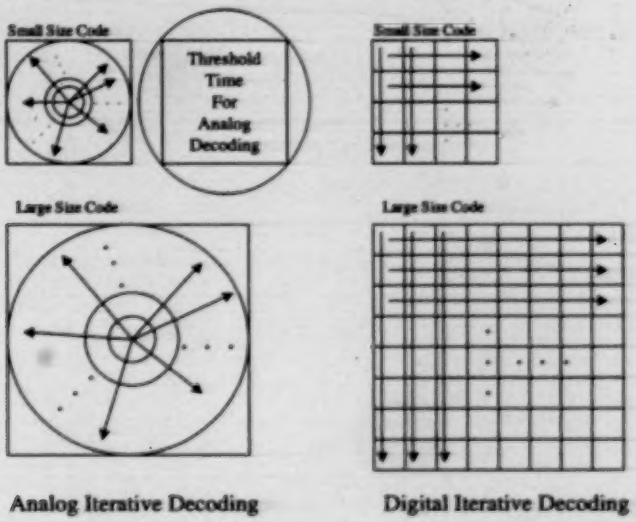

Figure 2.3: Analog iterative decoding compared to digital iterative decoding

In a digital iterative decoding approach, information symbols are decoded by sweeping all rows and columns iteratively. However information is diffused in analog iterative decoding architectures. The diffusion time in analog decoders is limited by a constant threshold based on the physical characteristics of such circuits and signals. After this threshold the signals start converging to their final values. Thus for a large code analog decoding can be considered instead of digital decod-

ing. However for a small code digital decoders perform faster than analog decoders due to the intrinsic threshold delay of analog decoders [1]. These two different approaches are illustrated in Fig. 2.3. For small size codes the intrinsic threshold time of analog iterative decoding is longer than the sweeping time of the digital design architecture to stabilize the values. However for large sized codes analog iterative decoder's diffusion time is shorter than the sweeping time of the digital iterative decoder to settle down.

Fully Differential Circuit Architecture

Another beneficial feature of analog implementation is a fully differential architecture. That is, signals are transmitted as the relative proportion of two or more analog currents, instead of an absolute value on a single wire. Since the effect of many error-causing events such as signal interference or device imperfection is equal for all signals, differential processing will eliminate them. Even though common mode disturbances can be remarkably large, the differential values remain unchanged and the signals containing information remain reliable [1].

Now we briefly explain the basic topology of analog circuits used in analog iterative decoders and point out their characteristics, for instance, its operation region and power consumption.

Translinear Circuits and Logarithmic Characteristics

The translinear circuit is the basic building block of analog decoder. The word *translinear* stands for describing exponential current-voltage characteristic of circuits such as bipolar transistors or CMOS transistors in the subthreshold region. In such a circuit voltages can be represented as a logarithmic function of currents, and by *trans*lation of nonlinear operation into the *linear* one, using *translinear* circuit, we can easily implement nonlinear operations. In particular, to multiply two input currents it is possible to add their corresponding voltages due to the logarithmic characteristic of the translinear device [23].

Moreover, the logarithmic domain gives us a large dynamic range of operation and makes translinear circuits suitable to implement traditional digital computation

circuits requiring massive multiplications with wide ranges of operation [23].

CMOS Cell in Saturated Weak Inversion Region

The CMOS transistor can be an element to implement a translinear circuit according to its intrinsic physical characteristic under the saturated weak inversion condition expressed by [1]

$$I_D = I_S \cdot e^{\frac{V_{gs}}{n V_T}} \cdot [1 - e^{\frac{-V_{ds}}{V_T}}]. \tag{2.23}$$

where I_D is the device current. Weak inversion, or subthreshold, is the condition when the gate voltage is less than the threshold voltage of the device, which is a constant process parameter i.e., $V_{gs} < V_{th}$, and I_D is less than $\frac{1}{10}$ of I_S, the device specific current defined by physical parameters of device. Saturation occurs when $V_{ds} > V_{gs}$, and under this condition we can neglect the second term of Equ. 2.23 to extract the basic equation expressed in Equ. 2.24 to build a translinear circuit [1].

$$I_D \propto e^{Con.V_{gs}} \tag{2.24}$$

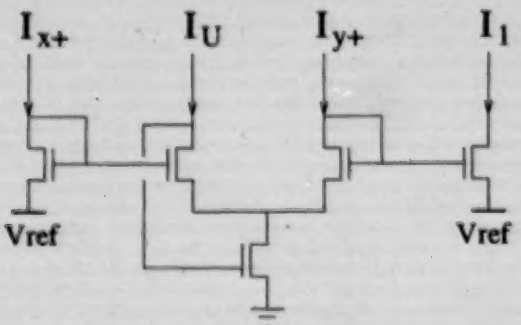

Figure 2.4: Basic block of analog differential multiplier with translinear loop [1].

The subthreshold current of CMOS transistors is very low, nano amperes for minimum size in $0.18\mu m$ CMOS technology, and comparing to BJT, (bipolar transistors), they do not have biasing currents. Therefore CMOS transistors are a good choice for a basic element of our low power analog circuit. For example Fig. 2.4 shows a basic multiplier which multiplies I_{x+} by I_{y+} and gives the normalized value of it to output, I_1. The analysis of this circuit is very simple. The 4 top cells in a row build a translinear loop. By starting from the left V_{ref} node and applying

Kirchhoff's Voltage Law (KVL) over the loop ending on the right V_{ref} node, we extract

$$I_{x+} \cdot I_{y+} = I_u \cdot I_1 \tag{2.25}$$

by using Equ. 2.24 for all cells in the loop. Dividing both sides of the equation above by a normalization current, I_u, we obtain our multiplication at the output:

$$I_{out} = \frac{I_{x+} \cdot I_{y+}}{I_u} \tag{2.26}$$

The 5th transistor at the bottom is for biasing the translinear loop.

Although we explained the translinear circuit concept used in analog decoders, we do not need to use them to build multipliers for the FFT. This is due to the fact that for the FFT we do not need two-input multipliers; All we need is a multiplication of an input with a constant, which we will discuss in the next Chapter.

Ultra Low Power Circuits

Because of the dynamic current due to the switching characteristic of digital circuits, the power consumption of analog circuits can be less than that of their digital counterparts. Not only is such a general rule remarkable in this case, but also the subthreshold operation region by itself means that the circuit works under very low current requirements and the power supply, V_{dd}, is less than that of normal CMOS circuits according to the fact that the gate voltage is usually under the threshold voltage for proper work.

In addition to these physical characteristics of analog circuits, we can change the topology of the basic multiplier circuits shown in Fig.2.4 to decrease the power consumption. The essential modification to this circuit to have less power consumption is to eliminate the V_{ref} based on the method proposed in [1].

2.5 Chapter Conclusion

In this Chapter, we motivated the design of an analog receiver based on recent analog decoder implementations. The goal is to take advantage of the low power consumption and the small required silicon area of analog circuitries to build an

entire analog receiver. The design of an analog FFT processor, which is the input interface between the RF front end and the decoder in our OFDM transmission scheme, is the main part of this work, which we will discuss in the next Chapter.

Chapter 3

Analog FFT Processor

By choosing OFDM as our transmission format, we have defined the type of input interface at the receiver front-end, which is the main part of this thesis. We need an interface to demodulate the received OFDM signal. As mentioned before, OFDM modulates information signals from the frequency domain to the time domain by using an IFFT to transmit over the communication channel and at receiver side we need to demodulate the signals by FFT processing. This leads to the design of an analog FFT processor. To build a low power receiver, the available digital FFT processors are not good choices since they need a very power hungry ADC running at high frequency, RF, and consume large silicon area which both directly increase the product cost [24, 25, 26].

3.1 Radix-2 FFT Butterfly Structure

3.1.1 8-FFT Structure

The butterfly diagram of an 8-FFT processor is depicted in Fig. 3.1 [27]. Dark circles on this graph represent the addition of the corresponding two input signals, white circles generate two copies of their single input, circles with -1 labeled inside are sign inverters and crossed circles are complex multipliers.

Each input in the butterfly graph is a complex differential value

$$x_k = (x_{ki+} - x_{ki-}) + j(x_{kq+} - x_{kq-}) \tag{3.1}$$

and the $W_N{}^k$s, FFT multiplicands, are complex constants on the unit circle

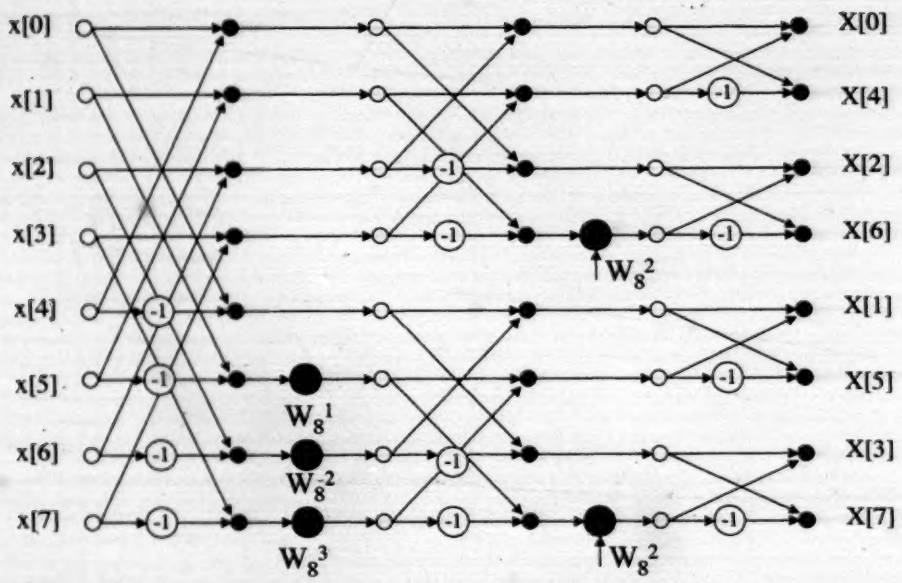

Figure 3.1: Butterfly Diagram of an 8-FFT. The inputs and outputs are complex differential values and $W_8{}^1$, $W_8{}^2$ and $W_8{}^3$ are complex constants [2].

$$W_N{}^k = e^{\frac{j2\pi k}{N}} = \cos\frac{2\pi k}{N} + j\sin\frac{2\pi k}{N} = WF_{ki} + jWF_{kq} \quad 1 \leq k \leq \frac{N}{2} - 1, \quad (3.2)$$

where N is the number of points in the FFT and the WF_{ki} and the WF_{kq} are the real and imaginary parts of the complex multipliers respectively. To make it simple from now on we call them Weighting Factors, WFs.

Modified WFs on Butterfly Diagram for 8-FFT

To have a better understanding of the WFs shown on the 8-FFT butterfly diagram we use the trigonometric circle view of them. In Fig. 3.2 one can see the circular view of all existing WFs in 8-FFT and their corresponding values. Since each input in our FFT structure contains four values to have complex differential values, we do not need to build each individual WF one by one. We can build some of the WFs and create the rest out of them by simply interchanging the positive/ negative input or/and the in-phase/quadrature input, which is in essence a rotation on trigonometric circle.

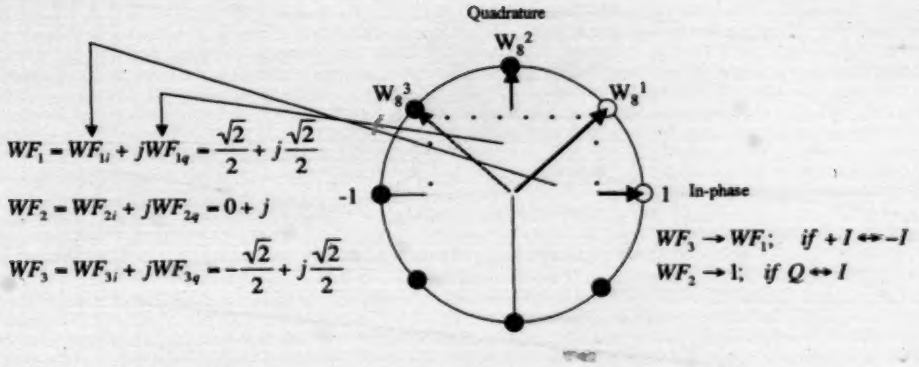
Quadrature
W_8^2
W_8^3
W_8^1
-1
1
In-phase
$WF_1 = WF_{1i} + jWF_{1q} = \frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2}$
$WF_2 = WF_{2i} + jWF_{2q} = 0 + j$
$WF_3 = WF_{3i} + jWF_{3q} = -\frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2}$
$WF_3 \rightarrow WF_1$; if $+I \leftrightarrow -I$
$WF_2 \rightarrow 1$; if $Q \leftrightarrow I$

Figure 3.2: 8-FFT Weighting Factors on trigonometric circle.

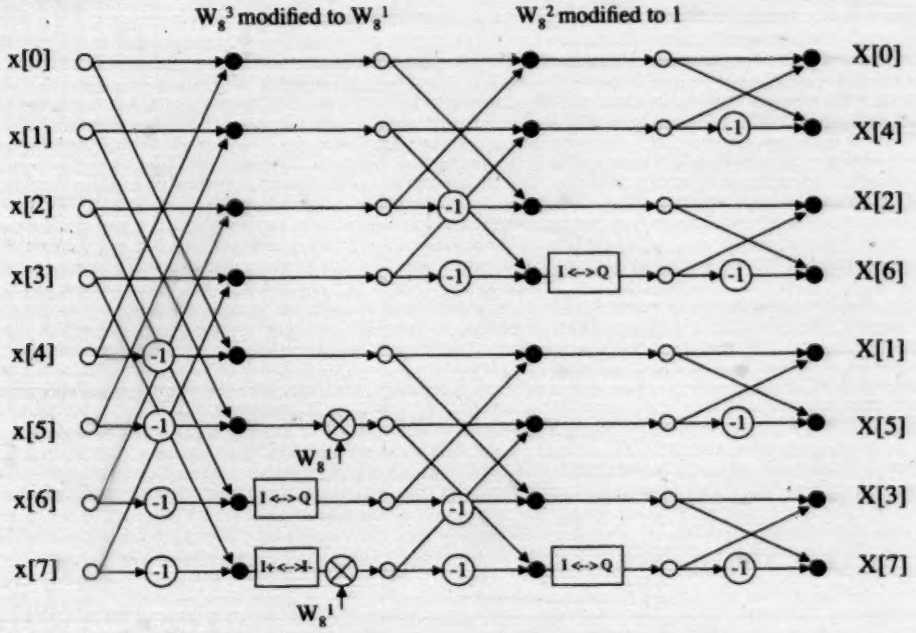

Figure 3.3: Modified Butterfly Diagram for 8-FFT .

For instance, as shown in Fig. 3.2, we can obtain WF_3 out of WF_1 by interchanging positive in-phase input and negative in-phase input or to create WF_2 we just need to interchange quadrature input and in-phase input to make it 1 and eliminate the multiplier of WF_2 in all over the 8-FFT diagram.

Doing the above modification, we build a new butterfly diagram for an 8-FFT as shown in Fig. 3.3

It is beneficial that we only need WF_1 for making an 8-FFT. We have simplified the FFT structure, eliminating three multiplier blocks out of five, without losing any accuracy on its computation just by judiciously using our complex differential values. It is obvious that we do not need any multiplier for sign inverters appearing in some parts of the diagram since we can use the same idea to achieve that.

To continue towards creating a 256-FFT it is instructive to look at the butterfly structure of FFT and find an algorithm to extend the 8-FFT to its larger version.

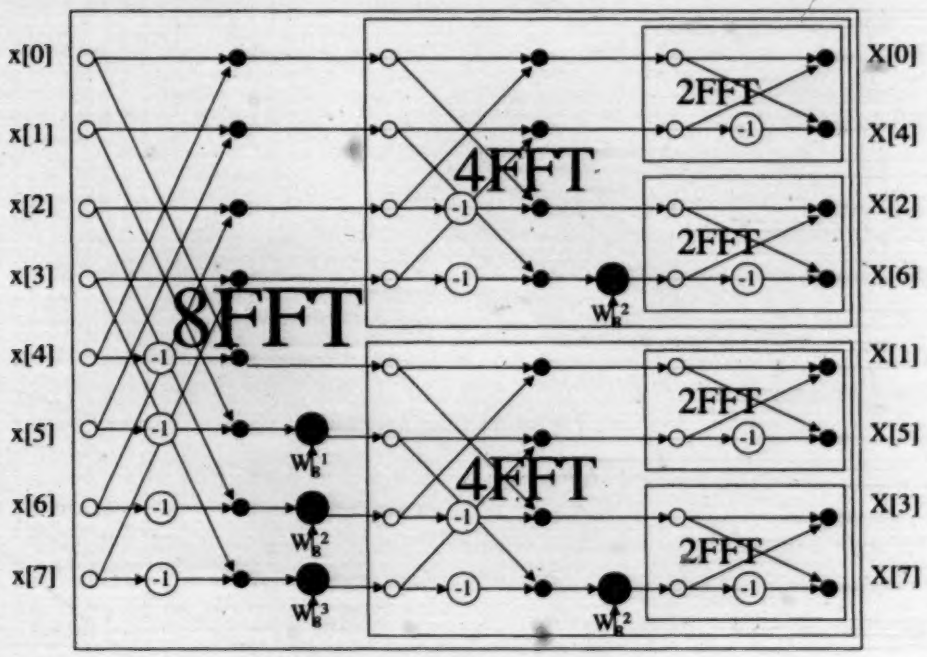

Figure 3.4: Butterfly Structure of FFT.

3.1.2 Extending Algorithm for Larger FFT

Fig. 3.4 is exactly the same as Fig. 3.1, only that here we want to show the butterfly structure of the FFT. Notice that to build a 4-FFT we have used two copies of a 2-FFT block, a multiplier and a few well-defined connections right at the input of these blocks. With the same argument we can obtain an 8-FFT as well. To find the algorithm to do that we divide each FFT block into upper half and lower half subsequences.

To build an N-FFT block using two $\frac{N}{2} - bit$ FFT blocks for the upper half part we need to add the inputs of these two blocks respectively and for the lower half part we need to subtract the inputs of the lower block from the inputs of the upper block respectively as well. After these summations and subtractions we use $\frac{N}{2} - 1$ multipliers based on Eq. 3.2 to create the WF_ks only for the lower half block to use in sequence as shown in Fig. 3.4. We will explain how to obtain these WFs in different stages on 256-FFT structure section later on. The next step is to explain how to sort the indices through the FFT butterfly diagram.

3.1.3 Index Sorting Algorithm

Input	Stages	m=1	m=2	m=3	Output
x[0] →	x[0]	x[0]	x[0]	x[0]	→ X[0]
x[1] →	x[1]	x[2]	x[4]	x[4]	→ X[4]
x[2] →	x[2]	x[4]	x[2]	x[2]	→ X[2]
x[3] →	x[3]	x[6]	x[6]	x[6]	→ X[6]
x[4] →	x[4]	x[1]	x[1]	x[1]	→ X[1]
x[5] →	x[5]	x[3]	x[5]	x[5]	→ X[5]
x[6] →	x[6]	x[5]	x[3]	x[3]	→ X[3]
x[7] →	x[7]	x[7]	x[7]	x[7]	→ X[7]

Figure 3.5: Index sorting algorithm for 8-FFT.

The idea is to split the inputs, as above, into two up and down groups in each

stage of the diagram. The number of stages is m for a 2^m-FFT. Starting with x[0], the first input, we choose every other input to make the up group, upper half block. Then starting with x[1], the second input, we do the same to make the down group, lower half block. Now we have two subsequences of even and odd indices. We continue doing this algorithm for each of the subsequences and so on to complete all the stages. For example if we consider the 2^3-FFT, which is the 8-FFT shown in Fig. 3.4, it has three stages to provide the output sequence. Through the graphical view of Fig. 3.5 one can follow the above algorithm to grasp it better [27].

Looking at the input and output columns in Fig. 3.5, one can show that some of the indices have remained in their initial positions after applying a sorting algorithm through all the stages, like 0, 2, 5, and 7. This might lead us to a smart way of looking at the algorithm, which will be easier to see when we use the binary representations of these indices. Fig. 3.6 shows the binary representations of both input and output indices. We notice that on the both sides the binary sequence of each output could be derived by reversing its corresponding input binary sequence. This is a well-known bit reversal step in the FFT algorithm and to find the different efficient bit reversal algorithms refer to [28]. Therefore we realize that the bit reversal of all 0, 2, 5, and 7 indices give us the same indices and that is why they remain on their initial place even after sorting.

Input	Binary Mirror	Output
x[**0**], **000**		**000**, X[**0**]
x[1], 001		100, X[4]
x[**2**], **010**		**010**, X[**2**]
x[3], 011		110, X[6]
x[4], 100		001, X[1]
x[**5**], **101**		**101**, X[**5**]
x[6], 110		011, X[3]
x[**7**], **111**		**111**, X[**7**]

Figure 3.6: Binary representations of 8-FFT's input and output, and their bit reversal dependency.

3.1.4 256-FFT Structure

In Fig. 3.7 we have extended the 8-FFT diagram to obtain a 256-FFT structure based on the extending algorithm explained above. One can see that the WFs of the 4-FFT and 8-FFT sub-blocks in this graph are exactly the same as those depicted in Fig. 3.1. It is fairly straightforward to follow the extending algorithm to generate any other large FFT.

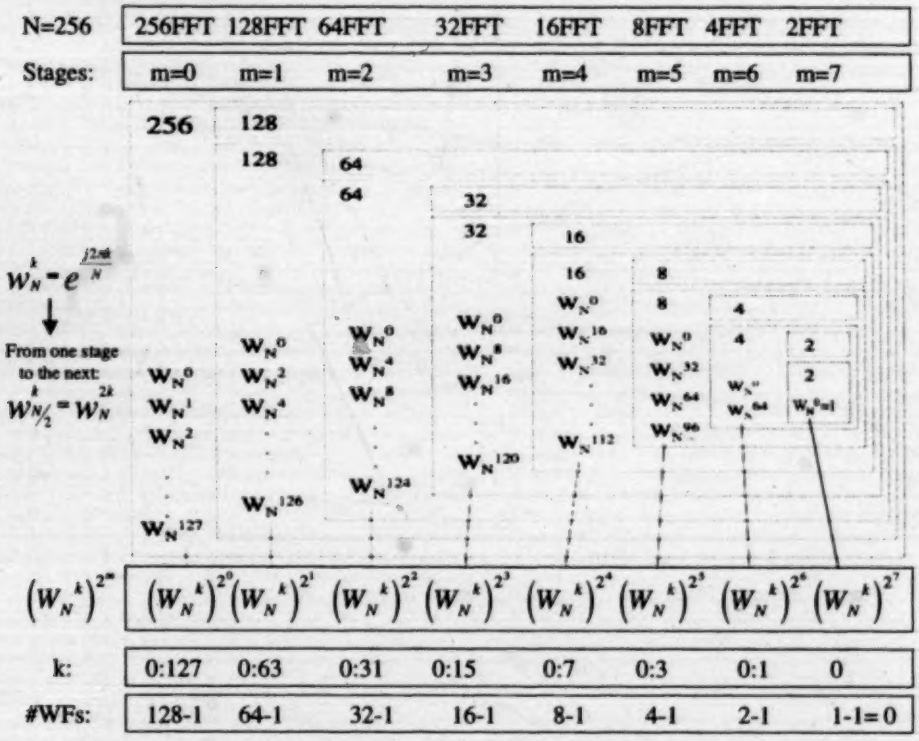

Figure 3.7: Generating Weighting Factors for different butterfly stages in the 256-FFT.

Now, the question is, what procedure can we use to generate all the WFs in different stages for such an extended 256-FFT. First of all, we briefly define the parameters mentioned in Fig. 3.7 and then explain the generating procedure for different WFs. The N is the number of points in the FFT, which is equal to 256 for our model. The m is the number of times we need to divide our block into the two upper and lower half groups based on butterfly structure to create 256-FFT using 2-FFT as a basic block. The W_N^ks are the complex values for Weighting Factors in

256-FFT based on Eq. 3.2. The k indicates the different WFs in each stage. For example in the 256-FFT we have 128 WFs with k starting from 0 to 127. Since, in each following stage we divide the inputs into the two half groups; we have 64 WFs in the 128-FFT and if we keep doing that we end up with 2 WFs for the 4-FFT and finally 1 WF for the 2-FFT. Actually the number of real WFs, shown as $\#WFs$ on the figure, is one less than the number of indicator k due to the fact that W_N^0 is always 1 and we do not need any multiplier to create a multiplicand that equals one. So, for instance, we do not have any multiplier for the 2-FFT and we have only one multiplier for the 4-FFT as we can see in both this figure and the previous figure, Fig. 3.1. Now is the time to explain the terminology $(W_N^k)^{2^m}$ we used in different stages on the figure and the procedure to generate WFs in different stages. As we mentioned above in each stage the number of WFs is half of the WF's number of previous stage since we divided the N-FFT into two $\frac{N}{2}$-FFTs. For example if we have W_N^k in the N-FFT for $0 \leq k \leq \frac{N}{2} - 1$, then the WFs in the $\frac{N}{2}$-FFT is $W_{\frac{N}{2}}^k$ for $0 \leq k \leq \frac{N}{4} - 1$. Also based on the Eq. 3.2 we can represent $W_{\frac{N}{2}}^k$ in the form of W_N^{2k} or equally $(W_N^k)^2$. This representation helps us to have our basic form of WFs, W_N^k, in all the stages. So after each division in every stage we use our basic form of W_N^k with the power of 2^m, to generate the WFs for all different stages as $(W_N^k)^{2^m}$. Therefore following this algorithm, we can easily create an N-FFT for any given N. Refer to [27] for more details and different mathematical approaches.

Modified WFs in the 256-FFT

The WFs on the last stage of the 256-FFT consist of 127 different values as shown in Fig. 3.7. These WFs have all other WFs in all other lower FFT stages. Therefore the maximum number of different weighting factors is 127. Due to the circular distribution of WFs on the unit circle and our complex differential signal model, as mentioned before we actually do not need to generate all different values of WFs shown in the 256-FFT diagram. We can interchange the in-phase/quadrature and/or positive/negative signals properly to reduce the number of different values for WFs by a factor of four, i.e., 32 different values for all WFs existing on 256-FFT. This idea has been illustrated graphically in Fig. 3.8. This graph contains 7

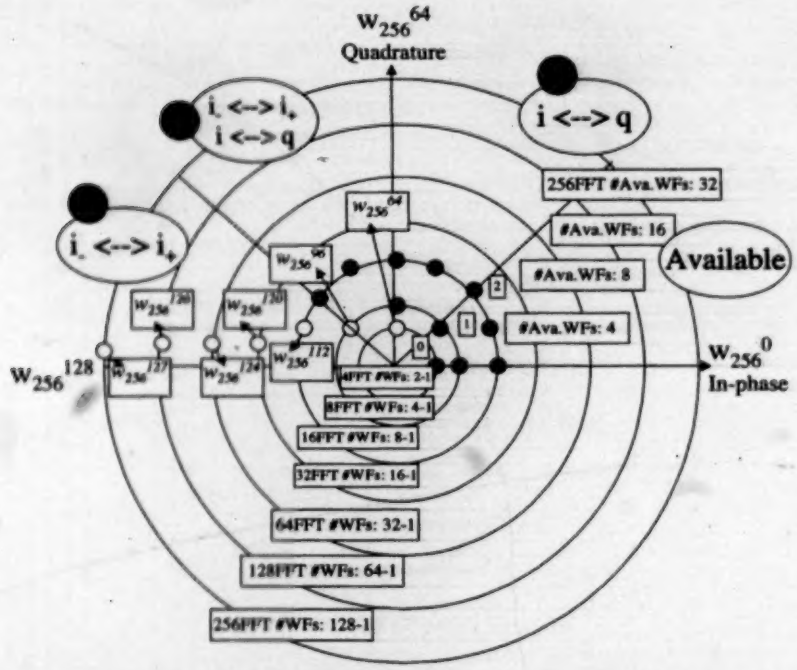

Figure 3.8: The circular view of the Weighting Factors on the unity circle.

circles representing specific FFT stages from 4-FFT up to 256-FFT. The number of WFs at each stage is displayed on the bottom of the graph. All the WFs exist on the top half of the unit circle. This area is divided into four different regions: available and three other regions called 1, 2, and 3. The only WFs that exist in available region are the ones we need to generate and the rest can be created from these by properly interchanging the in-phase/quadrature and/or positive/negative signals as illustrated for each region on this graph. For example, if we consider the last WF on the third circle, related to 16-FFT, which is represented as a white dot $W_{256}{}^{112}$, it is equal to $W_{16}{}^{7}$ due to the previous discussion by dividing both N and k by factor 16 to get WF for stage 16-FFT. Also $W_{16}{}^{7}$ can be created from $W_{16}{}^{1}$ by interchanging positive in-phase signal with negative in-phase signal as we can also see it graphically. Therefore the number of WFs in the available region is decreased by a factor of 4 as illustrated in this graph.

3.2 Expandable Analog FFT Circuit Design

Looking at the FFT diagram shown in Fig. 3.1 we observe that besides copying operations, additions and multiplications by constant complex values, W^1, W^2 and W^3, are the only two operations required to create an FFT processor. It is well suited for implementation using analog CMOS circuits based on the fact that adding currents is just tying two wires together and weighting can be achieved by transistor sizing. Therefore the FFT block can be simply implemented in CMOS technology by using current mirrors as basic blocks.

In Fig. 3.1 all white dots are current mirrors without scaling, i.e. have the same W/L ratio. The dark crossed circles are complex multipliers which contain 4 current mirrors with different scaling factors to generate the complex multiplication of Eq. 3.1 and Eq. 3.2. The black dots are summation nodes realized by tying wires together. The current mirror shown in Fig. 3.9 can be used as a basic circuit for entire FFT. To use this circuit just for mirroring the scaling factor, WF, should be 1.

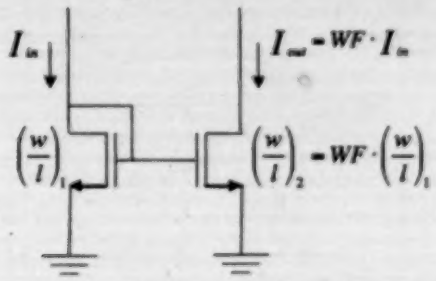

Figure 3.9: Current mirror: basic circuit to implement FFT.

Before going more in detail at the circuit level, first we need to translate our existing algorithm from the system level to electrical signals such as currents or voltages. In an analog FFT interface, to process the unit-less input values, we need to define an expression in terms of current and a constant with $\frac{1}{Amp}$ dimension as follow:

$$x[0] = [(I[0]_{xi+} - I[0]_{xi-}) + j(I[0]_{xq+} - I[0]_{xq-})].C \qquad (3.3)$$

Where: x[0] is unit-less input and C is constant $[\frac{1}{Amp}]$ that can be considered to be $\frac{1}{Amp}$ for now.

Also in terms of differential signals, we need to choose a biasing current so that all the positive and negative values can be represented by the circuit shown in Fig. 2.3. Because this topology allows us to have just positive currents, we define I_{bias} as follows to avoid having negative currents:

$$Min\{I[n]_{xi+}, I[n]_{xi-}\} = I_{bias} \tag{3.4}$$

$$Min\{I[n]_{xq+}, I[n]_{xq-}\} = I_{bias}. \tag{3.5}$$

It means:

If an absolute input value, ABSV, is positive:

$$I[n]_{xi-} = I_{bias}, I[n]_{xi+} = I_{bias} + ABSV, \tag{3.6}$$

otherwise:

$$I[n]_{xi+} = I_{bias}, I[n]_{xi-} = I_{bias} + ABSV. \tag{3.7}$$

Similarly, we define the quadrature parts.

3.2.1 Basic Circuit Block for Mirrors and Weighting Factors

In Fig. 3.4 each node consists four signals representing differential complex values. Translating it into the circuit level means that we have four wires for each node defined as I_{i+}, I_{i-}, I_{q+}, and I_{q-} shown in Fig. 3.10.

Since we always have two copies of each input nodes, represented as white nodes in Fig. 3.4, we have four inputs with two mirrors for each white node as a basic block for the FFT. The dark nodes do not have any circuit component since addition is tying two wires together. Therefore the basic block of the FFT has 12 transistors, 4 inputs and 8 outputs.

The only remaining block in the FFT structure is the multiplier which creates WFs. To create a differential complex multiplier using WF_i and WF_q shown in Fig. 3.2 we need to have a circuit to give us the four outputs of Eq. 3.8 as a differential complex values of the multiplier. We have four input signals and two weighting factors as follows:

$$((I_{i+} - I_{i-}) + J(I_{q+} - I_{q-})) \cdot (WF_i + J \cdot WF_q) :$$

$$\begin{aligned} Iout_{i+} &= I_{i+} \cdot WF_i + I_{q-} \cdot WF_q \\ Iout_{i-} &= I_{i-} \cdot WF_i + I_{q+} \cdot WF_q \\ Iout_{q+} &= I_{i+} \cdot WF_q + I_{q+} \cdot WF_i \\ Iout_{q-} &= I_{i-} \cdot WF_q + I_{q-} \cdot WF_i. \end{aligned} \tag{3.8}$$

As we see from the above equations, although the multiplier has only four outputs, it uses two mirrors per each output, and it means that we can also use the introduced basic block as a multiplier. We only need to change the transistor sizes to get different weighting factors.

3.2.2 NMOS/ PMOS Consecutive Stages

To build an FFT processor using the butterfly structure shown in Fig. 3.4 we need to use both NMOS and PMOS mirrors to let currents flows through the circuit. For example if we use NMOS mirrors as a basic block for building 2-FFT since we need to drive its inputs by PMOS transistors we can use PMOS mirrors for the 4-FFT and for the next stages we keep using NMOS and PMOS alternately.

3.2.3 Dummy Mirrors for Each WF Block

Since we have WF blocks in some nodes of each FFT larger than 8-bit, we need to think how we can use our NMOS/ PMOS consecutive topology having these extra blocks. In other words let us say in Fig. 3.4 we have a PMOS 4-FFT and we want to use NMOS mirrors to build an 8-FFT stage based on the above argument. What type of mirrors can we use for WF blocks, as not to change the direction of currents at the inputs of 4-FFT blocks? Since WF blocks are only in some places, not all nodes have multipliers in front, so we cannot use either of them without adding a dummy block at the end of each multiplier. So we add 4 mirrors of the opposite

FFT Size	Number of Transistors	
2-FFT	$2(FFTsize) \cdot 12(BasieBlock) =$	24
4-FFT	$4 \cdot 12 + 2(Butterfly) \cdot 24(PreviousStage) =$	96
8-FFT	$8 \cdot 12 + 2 \cdot 96 + 2(WFs) \cdot (12 + 8(dummy)) =$	328
16-FFT	$16 \cdot 12 + 2 \cdot 328 + 6 \cdot (12 + 8) =$	968
32-FFT	$32 \cdot 12 + 2 \cdot 968 + 10 \cdot (12 + 8) =$	2520
64-FFT	$64 \cdot 12 + 2 \cdot 2520 + 22 \cdot (12 + 8) =$	6248
128-FFT	$128 \cdot 12 + 2 \cdot 6248 + 46 \cdot (12 + 8) =$	14952
256-FFT	$256 \cdot 12 + 2 \cdot 14952 + 94 \cdot (12 + 8) =$	34856

Table 3.1: Number of transistors used in each FFT stage.

type, 8 transistors, at the end of each multiplier to make the current direction proper for the following stage.

3.2.4 Routing Algorithm for the Entire 256-FFT Circuit

Before explaining how we can route the FFT circuit, first it is worthwhile to have an idea about how big the circuit is and how many transistors are involved in each FFT stage on butterfly diagram shown in Fig. 3.7. The number of transistors used in each FFT stage is listed in Table 3.1. For a 2-FFT we have two basic blocks of 12 transistors; therefore the total number of transistors is 24 for a 2-FFT block. In the next stage, 4-FFT, there are four basic blocks right at the input of the stage in addition to two previous FFT blocks, 2-FFTs, based on the butterfly structure which give us the total number of 96 transistors for a 4-FFT block. For the other stages we can keep track of doing this plus the number of WFs that we have in each stage. Remember that the total number of each WF block is equal to 12, the number of transistors in a basic block, plus the number of transistors in a dummy mirror which is 8. In an 8-FFT block we have two WFs and it will go up in every larger FFT up to 94 which is the number of WFs in a 256-FFT block as represented in Table 3.1. Notice that the number of WFs in each stage of the FFT in the system diagram in Fig. 3.8 is higher than these in circuit. That is due to the modification we used in the circuit; the WFs with the quadrature values close to 1 could be created simply by interchanging the differential wires without using an extra multiplier. As shown the total number of transistors for a 256-FFT block is about 35000. To explain the

routing algorithm we focus on the 8-FFT shown in Fig. 3.3; however readers can easily use the argument here to extend it into any larger FFT structure.

Since we have 8 differential complex inputs, there are 32 input wires. to have an easy access to any of these wires we define the following terminology. We group all the four wires representing I_{i+}, I_{i-}, I_{q+}, and I_{q-} always in this sequence. It means that for instance if we want to reach the positive quadrature value of the first input, we can have a look at the third wire, or the fourteenth wire is the negative inphase value of the fourth input. In general we can follow these equations to find any input wire we are looking for:

$$I[n]_{i+} = 4n + 1, I[n]_{i-} = 4n + 2, I[n]_{q+} = 4n + 3, I[n]_{q-} = 4n + 4. \quad (3.9)$$

Also we have 8 basic blocks right at the input of the 8-FFT. To name the output of the basic blocks we divide them into two groups of upper half part and lower half part as we are already familiar from previous sections. The reason why we do that is the negative signs existing in the lower half part which cause interchanging the differential wires at their output. We name all the outputs of the upper half part in the sequence proposed above and for the lower half part we use the same numbers to tie them together. Since we have two copies of each input signal in the butterfly structure, in the upper half we first name the all first copies and then all second copies in sequence.

For example consider the first and the fifth dark nodes in Fig. 3.3. The first input copy of those simply ties together and their second input copy ties together after interchanging the differential values of the lower basic block. All the individual wires of these can be found in the following Figures 3.10 and 3.11

This way of naming the wires gives us the advantage of using bus vectors for a long sequence of inputs whenever we do not need to change the order or after interchanging the wires. For instance, all the outputs of the upper half part after tying the corresponding pair will directly feed into the previous FFT block in butterfly diagram, 4-FFT here, simply by using a wire name like $Input < 1 : 16 >$.

The rest is to explain how we can build all different types of WFs and name

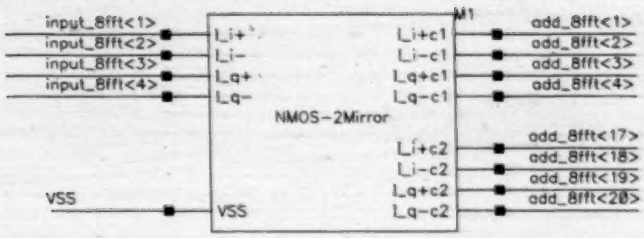

Figure 3.10: First basic block of the upper half part of the 8-FFT stage.

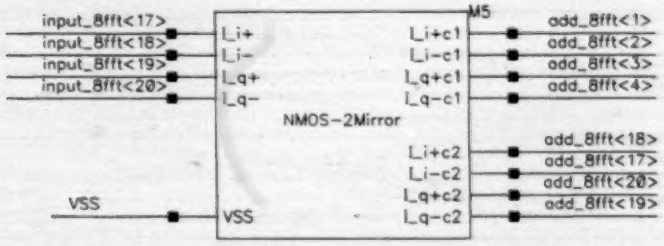

Figure 3.11: First basic block of the lower half part of the 8-FFT stage.

them properly. In Fig. 3.2 we have three types of WF called W_8^1, W_8^2 and W_8^3. They can be considered as a midpoint of a region for larger FFTs which have more WF and readers can apply the argument here for larger FFTs.

For the region of W_8^1 we can use the multiplier structure we explained early on using Eq. 3.8. In the second region related to W_8^2 we use the same argument with knowledge of $WF_i = 0$ and $WF_q = 1$. It gives us the way to interchange the in-phase and quadrature wires correctly as follows:

$$((I_{i+} - I_{i-}) + J(I_{q+} - I_{q-})) \cdot J \cdot WF_q :$$

$$\begin{aligned} Iout_{i+} &= I_{q-} \\ Iout_{i-} &= I_{q+} \\ Iout_{q+} &= I_{i+} \\ Iout_{q-} &= I_{i-}. \end{aligned} \tag{3.10}$$

From a circuit's point of view, it means the wire interchanging of the I(n) differential complex group like

$$I[n]_{i+} = 4n+4, I[n]_{i-} = 4n+3, I[n]_{q+} = 4n+1, I[n]_{q-} = 4n+2. \quad (3.11)$$

The final region is related to $W_8{}^3 = -WF_i + J \cdot WF_q$. It means that we need to create a multiplier block to give the same output as the following Eq. 3.12.

$$((I_{i+} - I_{i-}) + J(I_{q+} - I_{q-})) \cdot (-WF_i + J \cdot WF_q) :$$

$$\begin{aligned} Iout_{i+} &= I_{i-} \cdot WF_i + I_{q-} \cdot WF_q \\ Iout_{i-} &= I_{i+} \cdot WF_i + I_{q+} \cdot WF_q \\ Iout_{q+} &= I_{i+} \cdot WF_q + I_{q-} \cdot WF_i \\ Iout_{q-} &= I_{i-} \cdot WF_q + I_{q+} \cdot WF_i. \end{aligned} \quad (3.12)$$

Again if the $-WF_i = -1$ and $WF_q = 0$, the outputs only change their signs, interchanging the positive and negative wires.

3.3 Current Mirror Model

To model the FFT circuit behavior, we modeled the basic components: NMOS and PMOS current mirrors. To find the actual output of each mirror we simulated an NMOS mirror with a PMOS diode connected load and a PMOS mirror with an NMOS diode connected load shown in Fig. 3.12. The reason that we can use such a model for the entire FFT circuit is due to the fact that if we tie a second current mirror at the output node it is the same as if we increase the first current by that new current. The loading effect of the second mirror does not change the mirror characteristic since we have a diode connected load as a next stage everywhere in the FFT. We also checked this behavior by simulations. All the other nodes on the circuit are isolated from this output node through the gate of the loads.

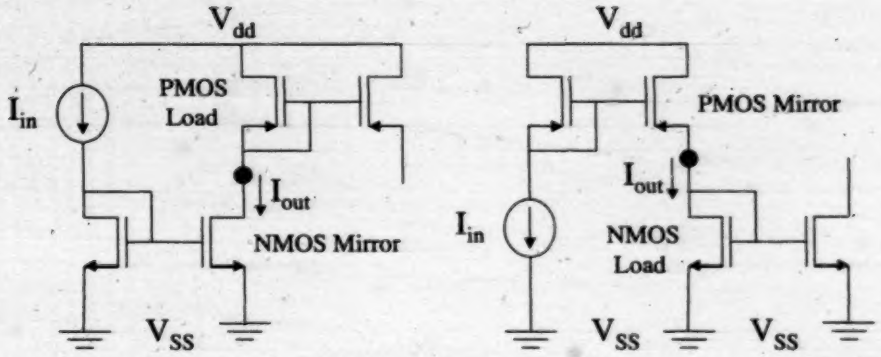

Figure 3.12: Basic model to find the mirror characteristic over the large input range.

The DC parameters and the I_{out} versus I_{in} characteristic of both the NMOS current mirror with PMOS load and the PMOS current mirror with NMOS load are listed in Fig. 3.2 and Fig. 3.3 respectively. Different operation regions can be realized by increasing the input current. By comparing the corresponding V_{ds} and V_{dsat} in each row we can define the saturation/ triode region and through comparing V_{gs} and V_{th} we can define the weak / strong inversion region.

Since by increasing the input current, V_{ds} get decreased and V_{gs} get increased, we face three different regions. First we are on weak inversion / saturation region since $V_{gs} << V_{th}$ and $V_{ds} > V_{gs} - V_{th}$. Then we have strong inversion / saturation region, since $V_{gs} > V_{th}$ and still $V_{ds} > V_{gs} - V_{th}$. The last region is strong inversion / triode region, since $V_{gs} > V_{th}$ and $V_{ds} < V_{gs} - V_{th}$. All these regions are plotted in Fig. 3.13. However the data for last region are not provided on these tables since we do not want to operate in triode region.

To consider the actual behavior of the mirrors based on the results given above we fitted a linear curve to them in the log domain and used these curves in our Matlab code. As it is clear in Fig. 3.13 the actual mirror's outputs are slightly larger than the inputs, shown as a line on the plot. It is mostly due to different values of V_{ds} on each branch of the mirror. For example if we consider an NMOS mirror with PMOS diode connected load shown in Fig. 3.12, the input transistor has a V_{ds} equal its V_{gs} which is typically smaller than the output transistor's V_{ds}, V_{dd} minus load V_{gs}.

$I_{in}(nA)$	$I_{out}(nA)$	$V_{ds}(V)$	$V_{dsat} = V_{gs} - V_{th}(mV)$	$V_{th} = 475(mV)$ $V_{gs}(mV)$
0.1	0.14	1.67	43	67
0.5	0.66	1.62	44	119
1	1.29	1.59	44	142
5	6.3	1.54	45	196
10	12.42	1.52	45	219
50	60	1.46	45	276
100	119	1.43	47	302
500	581	1.36	49	366
1000	1143	1.32	58	396
5000	5490	1.22	75	473
10,000	10690	1.15	93	512
50,000	51,000	0.862	173	637
100,000	99,300	0.642	233	723

Table 3.2: NMOS mirror characteristic over the wide input/output range.

$I_{in}(nA)$	$I_{out}(nA)$	$V_{ds}(V)$	$V_{dsat} = V_{gs} - V_{th}(mV)$	$V_{th} = 475(mV)$ $V_{gs}(mV)$
0.1	0.13	1.72	39	117
0.5	0.63	1.67	39	140
1	1.24	1.65	40	193
5	6.1	1.59	40	241
10	12.08	1.57	40	272
50	59	1.51	42	334
100	116.7	1.49	46	362
500	564	1.43	53	430
1000	1108	1.4	75	469
5000	5314	1.33	130	583
10,000	10460	1.29	185	645
50,000	50,610	1.17	375	916
100,000	99,570	1.08	545	1160

Table 3.3: PMOS mirror characteristic over the wide input/output range.

A Linear curve for PMOS mirror is slightly closer to the ideal mirror since a PMOS transistor has smaller mobility and it causes less difference in the values of V_{ds} on their mirror branches.

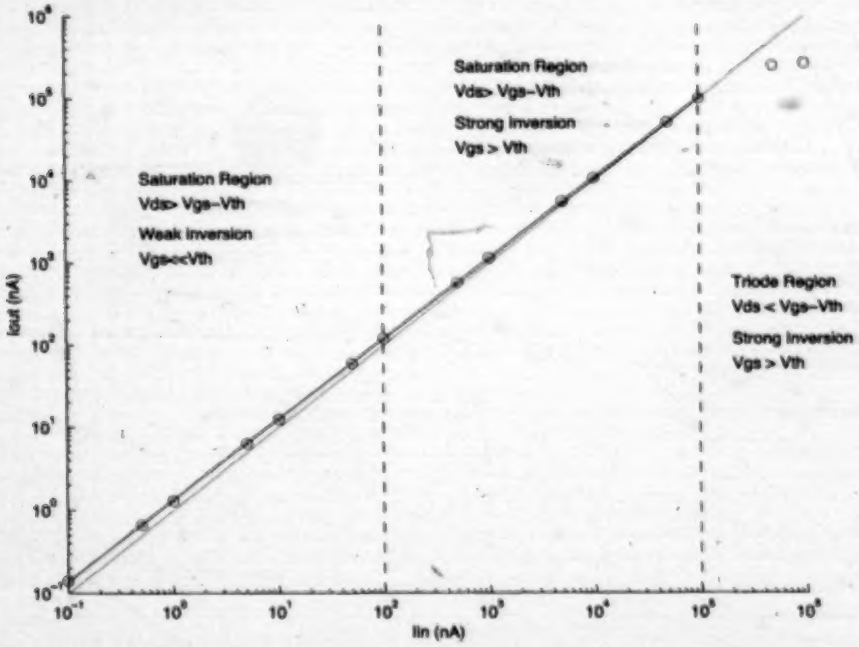

Figure 3.13: I_{out} versus I_{in} characteristic of NMOS current mirror with PMOS load.

3.3.1 Current Scaling at Every FFT Stage

Since we add currents in each FFT stages, the absolute value of currents get bigger and bigger in each stage when we go further in butterfly structure. This will result in more power consumption and more nonlinearity as well. We scale the $\frac{W}{L}$ of the output transistor of each current mirror participating in addition, white circles shown in butterfly diagram in Fig. 3.1, in every stage of FFT to get the same input/output current range after each addition. We chose the 0.45 as a scaling factor to make the absolute amount of output current in the same range of those of inputs. We discuss the impact of such a modification in greater detail in the simulation chapter.

After scaling transistors, since we have roughly the same input output current range for any N-FFT block, we can define and choose a specific bias current in the

range of our operation, from $100(pA)$ to $100(uA)$, represented in Fig. 3.13. Having an operational bias current for our FFT block makes it possible to compensate the existing offset on linear curve fitted in Fig. 3.13 by again slightly changing $\frac{W}{L}$ of the output transistor of each current mirror participating in addition.

3.4 Mismatch model

There are many considerations to take into account in analog design such as device mismatch, body effect and channel-length modulation. However most of the error due to channel-length modulation and body effect is common-mode. Thus by using a fully differential circuit, just the differential errors occur. On the other hand, mismatch for small devices can have a significant impact to cause an error. Based on the device characteristics, the magnitude of mismatch is approximately inversely proportional to device area. Therefore the consideration of device sizing is important at this point.

First we modeled the mismatch for each transistor pair due to their threshold voltage, V_{th}, in each current mirror as an additive white Gaussian random variable and we checked the sensitivity of the FFT due to the different values of the variance of this random variable.

Also, to analyze the impact of mismatch over the entire N-FFT block, we model the mismatch of all inside nodes of FFT block as an additive input referred mismatch source, shown in Fig. 3.20, which is also a normal random variable with calculated variance. Both models produce the same BER simulation performance which we discuss in the simulation results section.

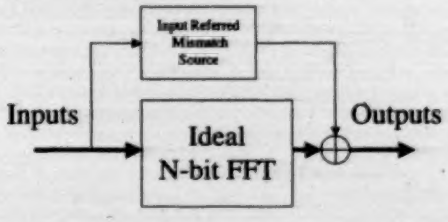

Figure 3.14: Input Referred Mismatch model for the N-FFT block .

3.4.1 Mathematical View for Mismatch Model

We explain our mismatch model both for strong inversion and subthreshold region and compare them.

Mismatch for Strong Inversion

The threshold voltage, V_{th}, variation is the dominant source of mismatch compared to the other sources like (W/L) or K which are physical parameters [29]. The V_{th} variation can be modeled as a normally distributed random variable with zero mean and variance of δ_ϵ^2 [29, 30]. We assume ΔV_{th} as a V_{th} variation for I_D in saturated strong inversion as expressed by

$$I_D = (\frac{W}{L})K(V_{gs} - V_{th} - \Delta V_{th})^2. \tag{3.13}$$

If we expand this equation we will obtain Eq. 3.14

$$I_D = (\frac{W}{L})K(V_{gs} - V_{th})^2 - 2 \cdot (\frac{W}{L})K(V_{gs} - V_{th})\Delta V_{th} + (\frac{W}{L})K\Delta V_{th}^2. \tag{3.14}$$

The first term is our ideal current, and the second part is the dominant mismatch part since the last term can be ignored due to small variance of ΔV_{th}^2. We rewrite it as follows:

$$I_D = I_{ideal} \cdot (1 - \frac{2 \cdot \Delta V_{th}}{V_{gs} - V_{th}}). \tag{3.15}$$

ΔV_{th} as we mentioned above can be represented as $V_{th} \cdot \varepsilon$ where ε is a normal distributed random variable, $\varepsilon : N(0, \delta_\epsilon^2)$. Therefore the Equ. 3.15 turns to the following equation:

$$I_D = I_{ideal} \cdot (1 + Con_{str} \cdot \varepsilon), \tag{3.16}$$

where Con_{str} is a normalizing factor in strong inversion equal to $\frac{2V_{th}}{V_{gs}-V_{th}}$. By choosing $V_{gs} = 2V_{th}$ this normalizing factor becomes 2. Therefore we acquire a normally distributed additive mismatch term for V_{th}.

To have an idea about the order of its variance, δ_ε^2, we choose a threshold voltage variation, ΔV_{th}, equal to ± 45 (mV), hence ε is 10% for typical value of $V_{th} = 450(mV)$ in our process technology, 180 nm [31]. Since for normal distribution we can assume that 97% of our outcomes are less than the three time of the standard variation, $3 \cdot \delta_\varepsilon$, so the δ_ε is about 0.033 and δ_ε^2 is equal to 0.001 times Con_{str}^2, normalizing factor.

Mismatch for Subthreshold (Weak Inversion)

To model the mismatch in subthreshold we look at the current equation given in Eq. 2.23.

$$I_D = k' \cdot e^{\frac{V_{gs}}{n \cdot V_T}} \cdot [1 - e^{\frac{-V_{ds}}{V_T}}]. \tag{3.17}$$

The k' is a process parameter which is exponentially dependent on V_{th}, V_T is the temperature coefficient known as kT/q. For saturation region we can neglect the V_{ds} term since V_{ds} is higher than $100(mV), 4kT/q$, to obtain the following equation [32].

$$I_D = k' \cdot e^{\frac{V_{gs}}{n \cdot V_T}}, \tag{3.18}$$

where k' is in form of $e^{\frac{-V_{th}}{n \cdot V_T}}$, so we can rewrite the Eq. 3.18 as follows:

$$I_D \propto e^{\frac{-V_{th}-\Delta V_{th}}{n \cdot V_T}} \cdot e^{\frac{V_{gs}}{n \cdot V_T}}; \tag{3.19}$$

$$I_D = I_{ideal} \cdot e^{\frac{-\Delta V_{th}}{n \cdot V_T}}. \tag{3.20}$$

As we mentioned in previous section ΔV_{th} can be represented as $V_{th} \cdot \varepsilon$ where ϵ is a normal distributed random variable, $\varepsilon : N(0, \delta_\varepsilon^2)$, therefore $e^{Con_{sub} \cdot \epsilon}$ is a log-normal random variable as a mismatch source of transistor pairs in weak inversion region where Con_{sub}, subthreshold normalizing factor, is equal to $\frac{V_{th}}{V_T}$. This normalizing factor, for $V_{th} = 450(mV)$and $V_T = 25(mV)$, is about an order of magnitude higher than Con_{str}, strong inversion normalizing factor.

If we expand the Taylor series of $e^{Con_{sub} \cdot \epsilon}$ in Eq. 3.20 we obtain

$$I_D = I_{ideal} \cdot (1 + Con_{sub} \cdot \varepsilon + \frac{Con_{sub}^2 \cdot \varepsilon^2}{2!} + \frac{Con_{sub}^3 \cdot \varepsilon^3}{3!} + ...) \qquad (3.21)$$

We can compare Eq. 3.16 for strong inversion with this equation. In addition to have all these higher order terms in Eq. 3.21, for subthreshold region, the order of magnitude difference between the normalizing factors Con_{str} and Con_{sub}, makes subthreshold operation much more sensitive to mismatch as we discuss in Chapter 4.

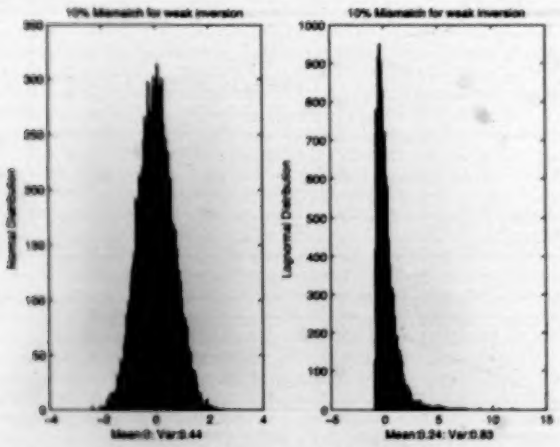

Figure 3.15: Log-normal distribution for weak inversion operation, 10% Mismatch. Its mean is a positive value 0.24 and its variance is about double

We also can see the effect of higher order terms in the Taylor series from probability studies where the log-normal distribution can be estimated by normal distribution for small values of variance and it starts to have log-normal distribution shape for larger values of variance. We depict these characteristics for different values of V_{th} mismatch in Fig. 3.15 up to Fig. 3.17. As we increase the mismatch from 1 % to 10 % the shape of the log-normal distribution tends to differ from normal, its mean deviates from zero, and its variance becomes larger. All these behaviors cause more impact in weak inversion compared to strong inversion. Again remember that this is not the major cause of the higher mismatch sensitivity in weak inversion compared to strong inversion. The major reason is due to the difference of their normalizing factor.

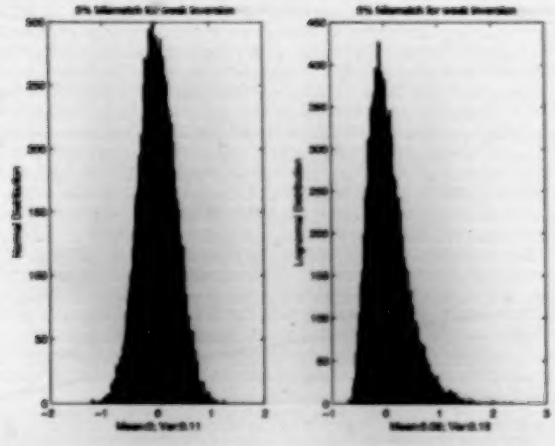

Figure 3.16: Log-normal distribution for weak inversion operation, 5% Mismatch. Log-normal distribution has small difference in mean and variance compared to normal distribution.

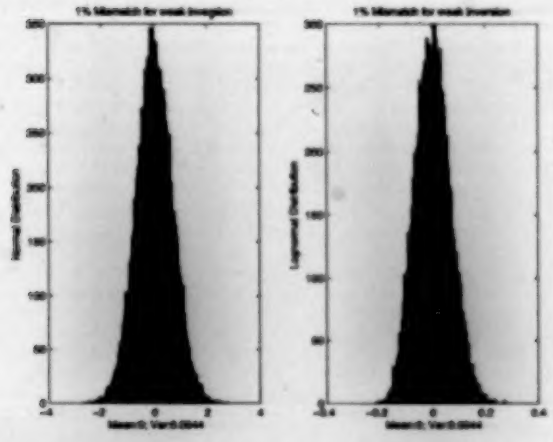

Figure 3.17: Log-normal distribution for weak inversion operation, 1% Mismatch. Log-normal distribution can be estimated by normal distribution.

3.4.2 Input Referred Mismatch Model

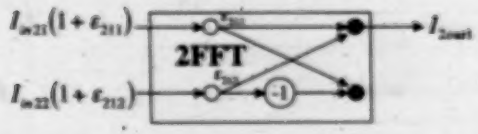

Figure 3.18: Mismatch for the 2-FFT block. Each current mirror has its ε_{sci} where s indicates the stage, c is either 1 or 2 representing the first or second copy of an input i.

For the 2-FFT we can model the mismatch as an external block as follow. Considering the mismatch of two current mirrors in the 2-FFT shown in Fig. 3.18, the following equation can be derived :

$$I_{out21} = I_{in21}(1+\varepsilon_{211}) + I_{in22}(1+\varepsilon_{212}) = \tag{3.22}$$
$$I_{in21} + I_{in22} + I_{in21}\varepsilon_{211} + I_{in22}\varepsilon_{212},$$

which can be seen as an ideal FFT plus an external mismatch term. If we substitute the input currents of 4-FFT coming into the 2-FFT on butterfly diagram shown in Fig. 3.19 we will find the same structure of the mismatch block for the 4-FFT as follows:

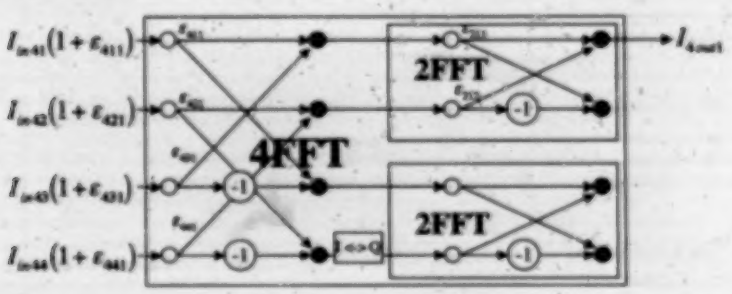

Figure 3.19: Mismatch for the 4-FFT block. Each current mirror has its ε_{sci} where s indicates the stage, c is either 1 or 2 representing the first or second copy of an input i.

$$I_{out1} = I_{in41}(1+\varepsilon_{411})(1+\varepsilon_{211}) + I_{in42}(1+\varepsilon_{412})(1+\varepsilon_{212}) +$$
$$I_{in43}(1+\varepsilon_{413})(1+\varepsilon_{211}) + I_{in44}(1+\varepsilon_{414})(1+\varepsilon_{212}) =$$
$$I_{in41}(1+\alpha_{41}) + I_{in42}(1+\alpha_{42}) + I_{in43}(1+\alpha_{43}) + I_{in44}(1+\alpha_{44}), \quad (3.23)$$

where α is in general in the form of $\varepsilon + \varepsilon' + \varepsilon \cdot \varepsilon'$. So the mismatch model for the 4-FFT has still the same structure as 2-FFT and the only differences are the coefficients in this model.

The input referred mismatch variance for the 4-FFT now can be derived from the above equations. In general for any I_{out4k} we have 4 statistically identical terms, $I_{ini}(1+\alpha_i)$, add together. Since the input signals are roughly in the same range, we can assume that statistically they are the same and we can say we have 4 times variance of α. Also α itself has roughly 2 times variance of each transistor mismatch, since the term $\varepsilon \cdot \varepsilon'$ is negligible. So the input variance of our mismatch model for the 4-FFT is roughly 8 times of that of each transistor pair.

We can use the same argument for the 8-FFT to obtain the input variance as follows:

$$I_{out8k} = \sum_{i=0}^{8} I_{ini}(1+\varepsilon'')(1+\alpha) = \sum_{i=0}^{8} I_{ini}(1+\varepsilon+\varepsilon'+\varepsilon''+\varepsilon\cdot\varepsilon'+\ldots+\varepsilon\cdot\varepsilon'\cdot\varepsilon'')$$

$$I_{out8k} = \sum_{i=0}^{8} I_{ini}(1+\beta). \quad (3.24)$$

However it is in the first stage that we have WFs involved. The effect of WFs is not just the mismatch of its transistor pairs. If we refer to Eq. 3.8, we will notice that each output of complex multiplier consist of two adding current together, so it produces double variance for each output. Therefore using the above argument for the upper half part of 8-FFT we obtain $8 \cdot 3\delta_\varepsilon^2$, and for the lower half part of 8-FFT we obtain $2 \cdot 8 \cdot 3\delta_\varepsilon^2$, so in total we have $3 \cdot 8 \cdot 3\delta_\varepsilon^2$ as an input referred mismatch variance for 8-FFT. We can keep doing this algorithm to find the overall variance of any larger FFT shown in Table. 3.4.

FFT Stages	Input Mismatch Variance	Radix-2 FFT
δ_2^2 :	$(1 \cdot 1) \cdot \delta_\varepsilon^2 =$	δ_ε^2
δ_4^2 :	$(4 \cdot 2) \cdot \delta_\varepsilon^2 =$	$8\delta_\varepsilon^2$
δ_8^2 :	$(8 \cdot 3) \cdot 3 \cdot \delta_\varepsilon^2 =$	$72\delta_\varepsilon^2$
δ_{16}^2 :	$(16 \cdot 4) \cdot 3 \cdot (3 \cdot 0.7) \cdot \delta_\varepsilon^2 =$	$403.2\delta_\varepsilon^2$
δ_{32}^2 :	$(32 \cdot 5) \cdot 3 \cdot (3 \cdot 0.7)^2 \cdot \delta_\varepsilon^2 =$	$2.1(10^3) \cdot \delta_\varepsilon^2$
δ_{64}^2 :	$(64 \cdot 6) \cdot 3 \cdot (3 \cdot 0.7)^3 \cdot \delta_\varepsilon^2 =$	$10.7(10^3) \cdot \delta_\varepsilon^2$
δ_{128}^2 :	$(128 \cdot 7) \cdot 3 \cdot (3 \cdot 0.7)^4 \cdot \delta_\varepsilon^2 =$	$52.3(10^3) \cdot \delta_\varepsilon^2$
δ_{256}^2 :	$(256 \cdot 8) \cdot 3 \cdot (3 \cdot 0.7)^5 \cdot \delta_\varepsilon^2 =$	$251(10^3) \cdot \delta_\varepsilon^2$

Table 3.4: Input referred mismatch variance for N-bit (radix-2) FFT from 2-FFT up to 256-FFT.

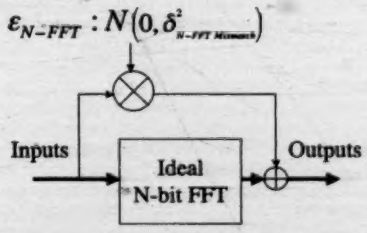

Figure 3.20: Input Referred Mismatch Variance for the N-FFT block .

Therefore we can generalize the input referred mismatch variance for an N-FFT as the following equation.

$$\delta_N^2 = (N \cdot m) \cdot 3 \cdot (3 \cdot AVE.WF)^{(m-3)} \cdot \delta_\varepsilon^2, \tag{3.25}$$

where m is the number of stages in butterfly structure, the first 3 is the average input factor for current stage due to WFs as explained above, the second factor 3 in this equation is the average input factor for previous stage due to WFs as well. The constant $AVE.WF$ is the average amount for WFs in every stages, which is 0.7. For m less than 4 the m-3 term should be 1 since there is on previous stage having WFs.

The reason that we do not consider $AVE.WF$ for 8-FFT, the first FFT stage with WFs, is due to the fact that the amount of currents in the lower half part of butterfly structure get increased because of addition inside the complex multiplier rather than those of upper half part. This factor is about 1.5 and can be cancelled out

FFT Stages	Radix-2 FFT	Radix-4 FFT	Radix-16 FFT
δ_2^2 :	$1 \cdot \delta_\varepsilon^2$	-	-
δ_4^2 :	$8 \cdot \delta_\varepsilon^2$	$4 \cdot \delta_\varepsilon^2$	-
δ_8^2 :	$(8 \cdot 3) \cdot 3 \cdot \delta_\varepsilon^2$	-	-
δ_{16}^2 :	$(16 \cdot 4) \cdot 3 \cdot 2.1 \cdot \delta_\varepsilon^2$	$(4^2 \cdot 2) \cdot 3 \cdot 2.1 \cdot \delta_\varepsilon^2$	$16 \cdot 3 \cdot 2.1 \cdot \delta_\varepsilon^2$
δ_{32}^2 :	$(32 \cdot 5) \cdot 3 \cdot 2.1^2 \cdot \delta_\varepsilon^2$	-	-
δ_{64}^2 :	$(64 \cdot 6) \cdot 3 \cdot 2.1^3 \cdot \delta_\varepsilon^2$	$(4^3 \cdot 3) \cdot 3 \cdot 2.1^2 \cdot \delta_\varepsilon^2$	-
δ_{128}^2 :	$(128 \cdot 7) \cdot 3 \cdot 2.1^4 \cdot \delta_\varepsilon^2$	-	-
δ_{256}^2 :	$(256 \cdot 8) \cdot 3 \cdot 2.1^5 \cdot \delta_\varepsilon^2$	$(4^4 \cdot 4) \cdot 3 \cdot 2.1^3 \cdot \delta_\varepsilon^2$	$(16^2 \cdot 2)3 \cdot 2.1^2 \cdot \delta_\varepsilon^2$

Table 3.5: Estimation of Input referred mismatch variance for different radix N-FFT structures.

by 0.7. Therefore the 8-FFT only has factor 3 as an average input factor and there is no 0.7, the average amount for WFs factor, for this stage. For the 16-FFT and larger, we take into account both the average input factor 3 and the average values of WFs 0.7 for each stage. This approximation works fairly good up to the 256-FFT. After several stages, to make our model better fitted, we need to again compensate the impact of increasing the amount of currents in the lower part of butterfly structure. For instance for the 256-FFT we change the last 0.7 factor to 0.9 in Eq. 3.25.

We can add the mismatch of each mirror inside the multipliers and each dummy transistor at their output, which are also normal random variable named ε_{wf} and ε_{dummy} respectively.

Looking at Table 3.4, we will find that the input referred mismatch variance is highly dependent on the number of stages and for example for the 256-FFT the variance is in order of 10^5 of the variance of a single transistor pair. So to make our N-FFT less sensitive to transistors mismatch one good idea is to decrease the number of stages in FFT butterfly structure. This fact leads us to use an FFT with a higher radix structure.

For instance, in a radix-4 FFT, at each stage we can add four currents rather than two currents, which we have done in our radix-2 FFT butterfly structure, since the addition of currents in analog domain costs nothing but tying wires together. By this type of change in the butterfly diagram, we obtain half the number of stages, it means only four stages for the 256-FFT. Also if we consider a radix-16 FFT we only

FFT Stages	Radix-2 FFT	Radix-4 FFT	Radix-16 FFT
δ_2^2 :	$1 \cdot \delta_\varepsilon^2$	-	-
δ_4^2 :	$8 \cdot \delta_\varepsilon^2$	$4 \cdot \delta_\varepsilon^2$	-
δ_8^2 :	$72 \cdot \delta_\varepsilon^2$	-	-
δ_{16}^2 :	$403.2 \cdot \delta_\varepsilon^2$	$201.6 \cdot \delta_\varepsilon^2$	$100.8 \cdot \delta_\varepsilon^2$
δ_{32}^2 :	$2.1 \cdot (10^3) \cdot \delta_\varepsilon^2$	-	-
δ_{64}^2 :	$10.7 \cdot (10^3) \cdot \delta_\varepsilon^2$	$2.55 \cdot (10^3) \cdot \delta_\varepsilon^2$	-
δ_{128}^2 :	$52.3 \cdot (10^3) \cdot \delta_\varepsilon^2$	-	-
δ_{256}^2 :	$251 \cdot (10^3) \cdot \delta_\varepsilon^2$	$28.4 \cdot (10^3) \cdot \delta_\varepsilon^2$	$6.76 \cdot (10^3) \cdot \delta_\varepsilon^2$

Table 3.6: Input referred mismatch variance comparison of N-FFT for radix-2, 4, and 16.

FFT Type	Input Mismatch Variance: 256-FFT	
Radix-2	$(2^8 \cdot 8) \cdot 3 \cdot 2.1^5 \cdot \delta_\varepsilon^2 =$	$251 \cdot (10^3) \cdot \delta_\varepsilon^2$
Radix-4	$(4^4 \cdot 4) \cdot 3 \cdot 2.1^3 \cdot \delta_\varepsilon^2 =$	$28.4 \cdot (10^3) \cdot \delta_\varepsilon^2$
Radix-16	$(16^2 \cdot 2) \cdot 3 \cdot 2.1^2 \cdot \delta_\varepsilon^2 =$	$6.76 \cdot (10^3) \cdot \delta_\varepsilon^2$
DFT	$(256^1 \cdot 1) \cdot 3 \cdot 2.1 \cdot \delta_\varepsilon^2 =$	$1.61 \cdot (10^3) \cdot \delta_\varepsilon^2$

Table 3.7: The comparison of the Input referred mismatch variances for different types of the 256-FFT.

need two stages to implement the 256-FFT. We have estimated the input referred mismatch variance for the radix-4 and the radix-16 FFT based on our radix-2 FFT and compared these in Table 3.5 and Table 3.6 respectively.

Based on this argument, we can also think of implementing a 256-FFT in one single step, which is implementing the DFT expressed in Eq. 2.7. We provide the input referred mismatch variances for these different structures in Table 3.7. As it is clear that the DFT has the least variance, its implementation is less sensitive to the mismatch of transistor pairs compared to other FFT structures. Using a DFT we obtain about 100 times less input referred mismatch variance, the impact of which will be discussed in Chapter 4. Also if we look at the number of current mirrors needed to build a 256-FFT using different radix structures, as shown in Table 3.8, we will notice that the number of current mirrors decreases as we increase the radix of the 256-FFT and hence decrease the number of stages. So there is a win-win senario in terms of the circuit complexity and the mismatch sensitivity to implement an analog DFT processor.

The win-win senario in terms of the circuit complexity and the mismatch sen-

256-FFT Type	Number of Current Mirrors/Output	
Radix-2	$2 \cdot (1+2+4+8+16+32+64+128) =$	510
Radix-4	$4 \cdot (1+4+16+64) =$	340
Radix-16	$16 \cdot (1+16) =$	272
DFT	$256 \cdot (1) =$	256

Table 3.8: The comparison of the number of current mirrors in 256-FFT for different radix structures.

sitivity for the analog DFT implementation is due to the different cost items that we face in digital versus analog FFT design. In a digital DFT implementation the addition is costly so we like to decrease the number of additions as much as possible. By sorting the inputs and using several stages we can minimize the number of additions and obtain the DFT, structure which is called FFT. Increasing the number of stages does not increase the cost of copying in digital implementation since we have registers and we can use our signals as many times as we need. However in an analog FFT implementation the copying of signals is a real cost since we need to use a current mirror to do so. Furthermore, there is no cost for additions in analog design. Therefore we need to decrease the number of stages to decrease the number of current mirrors as well. Clearly the FFT is not an optimum way to perform the DFT using analog circuitry. The DFT is better for an analog implementation due to its reduced number of current mirrors and its reduced sensitivity to mismatch compared to the radix-2 FFT structure.

Higher radix FFT structures, due to the large number of wires adding at one node, bring some design issues like large capacitances and output impedances. The large capacitance decreases the speed of the FFT. However, the speed of the receiver is limited by the speed of the iterative decoder. Also, we can obtain the proper output impedance by sizing the load transistors at each FFT stage.

3.5 Power Consumption

The possible current operating range is from 100 (pA) to 100 (uA) based on the current mirror model shown in Fig. 3.13. To have less power consumption we like to choose as low a bias current as possible. However based on the mismatch

argument we would like to operate in strong inversion. So we choose a low bias current in strong inversion by making the length of transistors larger. This also helps to minimize the mismatch due to minimum length.

We choose the bias current equal to 100 (nA) and the power supply, V_{dd}, for a 180-nm technology is 1.8 (V). So on average for a radix-2 256-FFT since we have 256 complex differential inputs, 1024 inputs, and 8 stages, the power consumption is on order of $1024 \cdot 8$ plus the number of multipliers in every stage times $I_{bias} \cdot V_{dd}$. The numbers of multipliers in the 256-FFT stage down to the 8-FFT stage is about $128 + 2 \cdot 64 + 4 \cdot 32 + 8 \cdot 16 + 16 \cdot 8 + 32 \cdot 4 = 768$. Therefore the total power for I_{bias} 100 (nA), and V_{dd} 1.8 (V) is in order of $(1024 \cdot 8 + 768) \cdot 18(nW)$ which is about 1.6 (mW).

3.6 Chapter Conclusion

In this Chapter, we presented a design methodology for an analog N-FFT using only current mirrors. We analyzed the mismatch of the current mirrors and we obtained an input referred mismatch model for the N-FFT. We showed that the radix-2 FFT structure has a very large input referred mismatch variance. The higher radix FFT structures have smaller input referred mismatch variances. Also, the number of current mirrors for the higher radix FFT structures is less than that of the radix-2 FFT structure. Hence although FFT is a good option for digital design, DFT is more proper for analog design. In the next Chapter we look at the BER performance of our system considering the design issues discussed in this Chapter.

Chapter 4

Simulations and System Performance

This chapter studies how we can estimate the performance of the analog FFT processor and what level of reliability we need. To answer these questions, first we should look at inputs and outputs as random variables and then compare input Signal to Noise Ratio (SNR) to output SNR of the FFT block.

The input signal, $y[n]$, consists of OFDM signals, $x[n]$, plus AWGN, n[n]: $N(0, \sigma^2)$ out of the channel.

$$y[n] = x[n] + n[n]. \tag{4.1}$$

Hence, the input SNR can be represented as:

$$Input_{SNR} = \frac{E[x^2[n]]}{E[n^2[n]]}. \tag{4.2}$$

The same approach is applicable for FFT output, we can assume:

$$Y[n] = X[n] + N[n]. \tag{4.3}$$

Where $Y[n]$ is analog FFT output, $X[n]$ is the actual FFT value and $N[n]$ represents all existing noise after the analog FFT processor including channel noise, circuit noise and any defect coming out of the analog FFT block. The output SNR is defined as:

$$Output_{SNR} = \frac{E[X^2[n]]}{E[N^2[n]]}. \tag{4.4}$$

Finally, we can find the performance of the analog FFT processor by comparing these two input and output SNRs and we need a design which does not decrease the performance of the existing decoder remarkably nor wastes complexity in designing the input interface to be too accurate.

4.1 Differential BPSK System Performance

Considering the BER vs. SNR performance, the noise power is doubled when using differentially coherent detection versus the original coherent technique [22].

The power performance of differential QPSK compared to common coherent PSK is about 2.3 dB worse at BER of 10^{-4}. However for differential BPSK it is less than 1 dB [22]. Theoretical BER vs. SNR performances are shown in Fig.4.1.

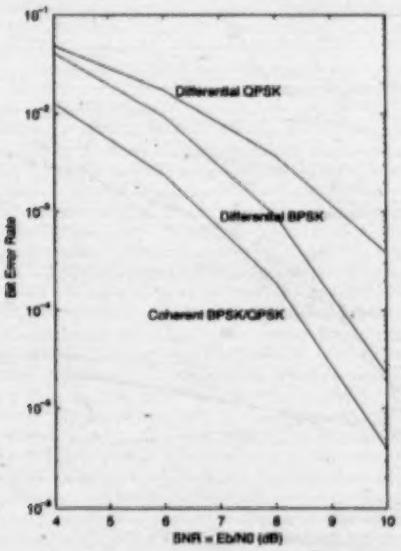

Figure 4.1: Ideal 256-FFT performance; Coherent demodulation compare to differential demodulation.

This probability of error for coherent BPSK/QPSK is the well-known Q function given by (4.5).

$$Coherent : P_b = Q\left(\sqrt{\frac{2E_b}{N_0}}\right). \tag{4.5}$$

The probability of error for differential coherent BPSK is expressed by

$$Dif.Coherent : P_b = \frac{1}{2}e^{\frac{-E_b}{N_0}}, \tag{4.6}$$

and for differential coherent QPSK the probability of error is based on the Marcum Q function and the modified Bessel function of order zero explained in [22].

In a spread spectrum scenario the limiting factor is interference of different users. Therefore by using higher order modulation people try to get higher spectral efficiency, in other words the differential QPSK in general is more spectrally efficient than differential BPSK. However in spread spectrum communications with a multiuser channel the spectral efficiency of N QPSK system is the same as 2N BPSK.

The advantage of using differential BPSK in this standard is to gain more in power efficiency while remaining the same in the spectral efficiency as compared to differential QPSK modulation.

We first simulated our ideal butterfly-structured FFT to check its functionality using the IFFT at the transmitter and AWGN channel. We obtained the same theoretical performance for this coherent BPSK model. Then we add a circular differential QPSK modulator and complex multiplier as our demodulator at transmitter and receiver respectively. This new result follows the theory as well. Both simulation results are shown in Fig.4.2.

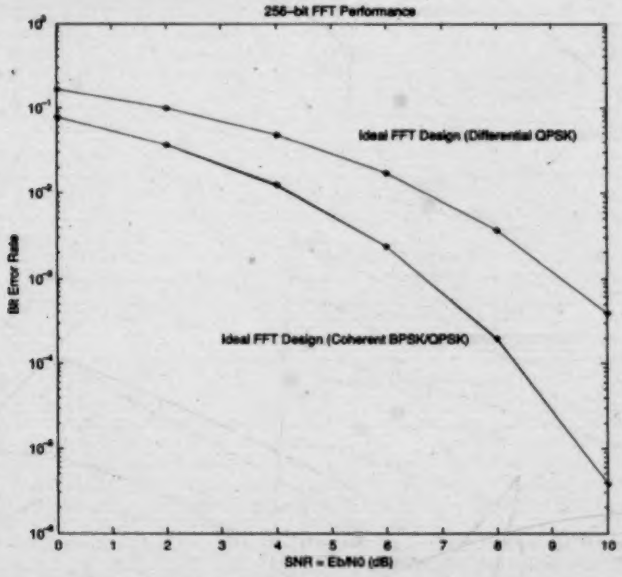
256-bit FFT Performance
Ideal FFT Design (Differential QPSK)
Ideal FFT Design (Coherent BPSK/QPSK)
Bit Error Rate
SNR = Eb/N0 (dB)

Figure 4.2: Ideal 256-FFT design for coherent versus differential design.

4.2 BER/SNR FFT Simulations

We simulated the OFDM communication system model shown in Fig. 2.2 using Matlab and a C program to characterize our FFT block considering different design issues as discussed in following sections. We ran the Monte Carlo simulation for 10^6 bit samples. We demonstrate the BER performance curve of the system versus $SNR = \frac{E_b}{N_0}$. First we look at the FFT performance without using decoder and then we explain how decoder can improve our performance.

4.2.1 Current Mirror Model for 256-FFT Simulations

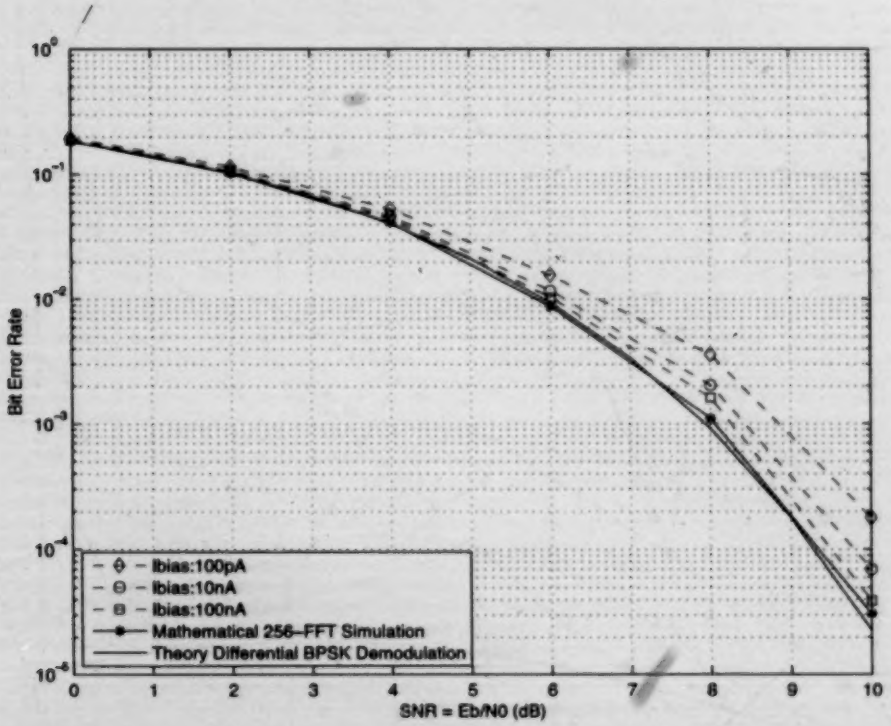

Figure 4.3: Statistical simulation of 256-FFT having current mirror model for different bias currents.

In this section we examine the performance of the 256-FFT using our current mirror linear curve model explained in previous chapter. We used this model at each current mirror and ran the statistical simulation for different bias currents to compare the 256-FFT performance with that of having exact current mirror, which

we call mathematical 256-FFT. The bias current range can be varied from $100(pA)$ to $100(uA)$ based on the model shown in Fig. 3.13. As it is clear in this figure, the more we increase the bias current the better curve fitting we have, in other words, the linear fit curve gets closer to the ideal line as we move from subthreshold into the strong inversion region. This behavior shows up in the statistical simulation as well shown in Fig. 4.3.

The loss in performance due to different bias currents can be mostly compensated by using the offset cancelation technique of the current mirror fitted linear curve, explained in mirror model section.

4.2.2 Current Scaling in every FFT Stage for 256-FFT Simulations

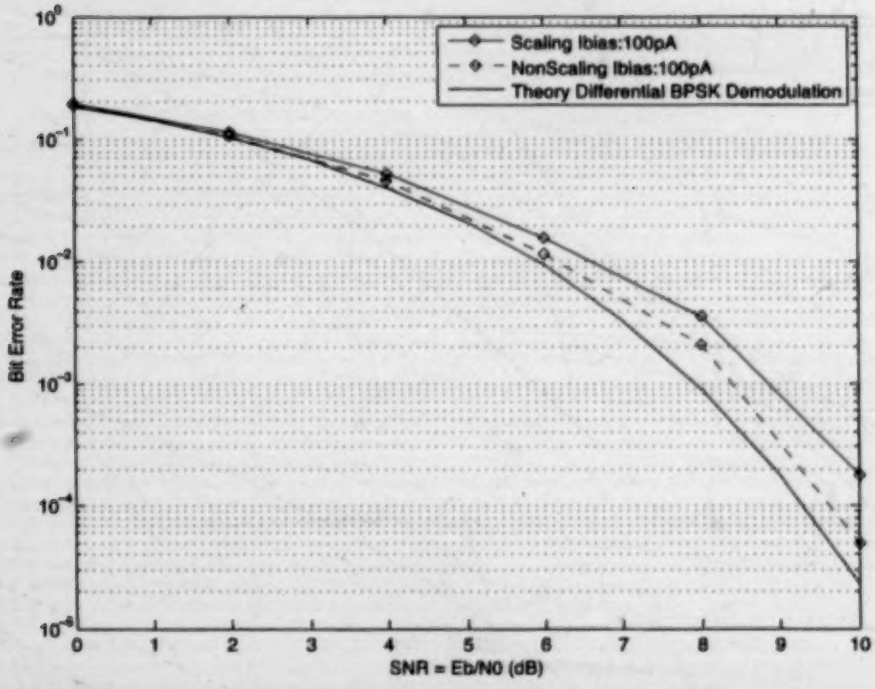

Figure 4.4: Statistical simulation of 256-FFT with scaling/non-scaling current at each stage for 100 (pA) bias current. The dashed line is a non-scaled version and the solid line is a scaled version.

We decided to scale the output mirror at each stage, based on the previous chap-

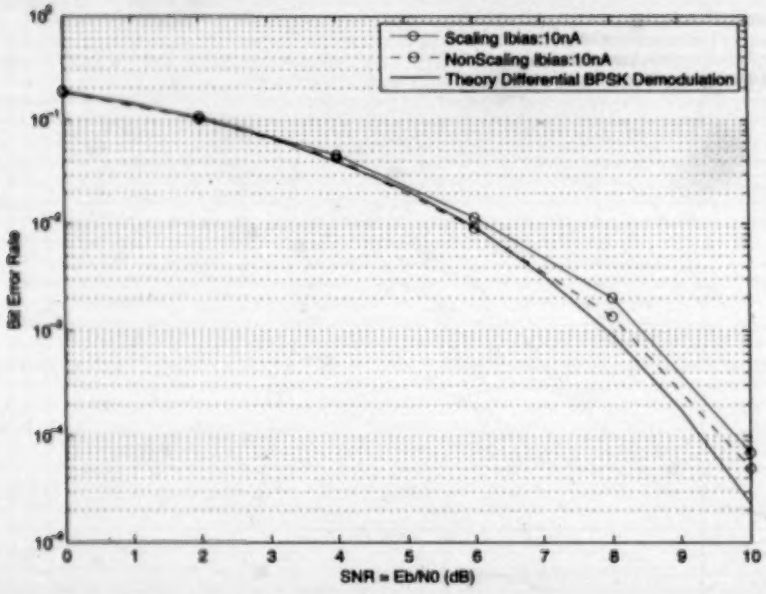

Figure 4.5: Statistical simulation of 256-FFT with scaling/non-scaling current at each stage for 10 (nA) bias current. The dashed line is a non-scaled version and the solid line is a scaled version.

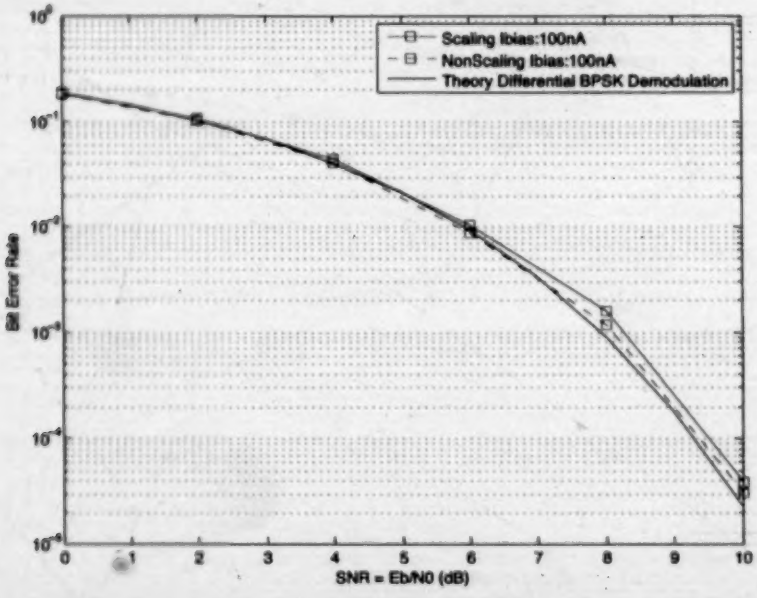

Figure 4.6: Statistical simulation of 256-FFT with scaling/non-scaling current at each stage for 100 (nA) bias current. The dashed line is a non-scaled version and the solid line is a scaled version.

ter explanation, to obtain roughly the same input output current range. We scaled the output of each adding mirror with a factor of 0.45 at each FFT stage. This scaling causes a degradation in the FFT performance for different bias currents. The following statistical simulations represented in Fig. 4.4, Fig. 4.5, and Fig. 4.6 show the impact of current scaling for 256-FFT.

Again, this loss in performance can be mostly compensated by using the offset cancelation technique of the current mirror fitted linear curve, explained in the current mirror model section (S. 3.3).

4.2.3 Simplification of WFs for 8-FFT up to 256-FFT Simulations

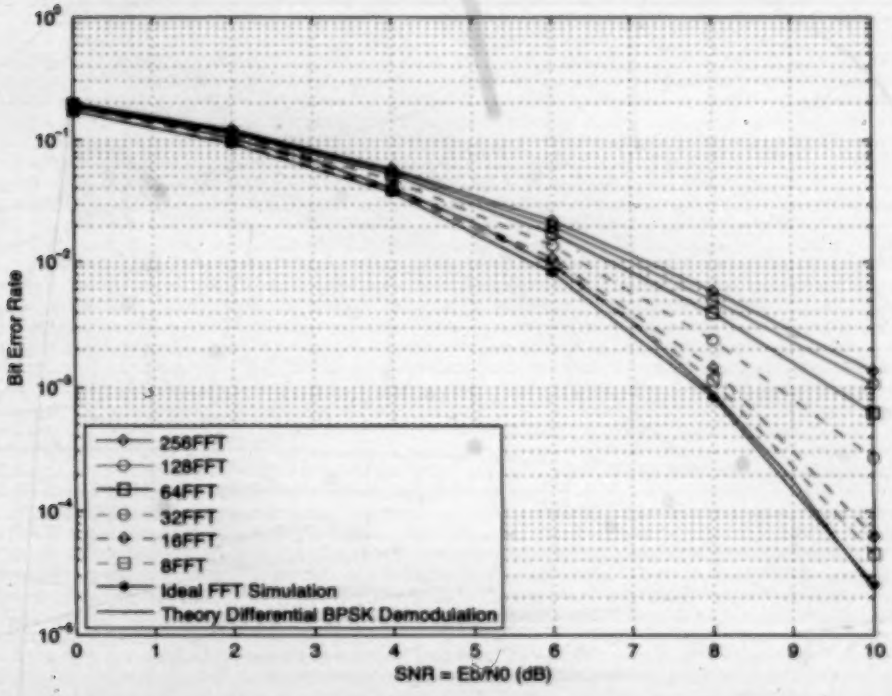

Figure 4.7: Statistical simulation of FFT, from 8 to 256 bit, with only three different values for WFs, 0.4, 07, and 0.9.

We simplified the FFT structure by using only 3 different values for WFs rather than 64 for all the complex multiplier inside FFT stages up to 256-FFT demonstrated in Fig. 3.8. The impact of such a simplification on the FFT BER perfor-

mance is illustrated in Fig. 4.7. The maximum loss of about 2 dB at BER of 10^{-3} is for the 256-FFT.

By such a simplification we do not save remarkable space on silicon area, because we still need complex multipliers at those nodes and the only difference is that the value of their WFs can be selected from three options rather than sixty four. Therefore by making WFs as precise as possible we compensate this 2 dB loss. One possible way to increase the number of WFs is to use higher number of fingers for making the width of transistors in layout design.

4.2.4 Mismatch Model for 8-FFT up to 256-FFT Simulations

In this section we discuss the sensitivity of the FFT for different mismatch variances of threshold voltage in transistor pairs. In the following simulations we examined two different models for mismatch. One is the FFT which has mismatch at each current mirror, shown as a solid line in all figure in this section, and the other one is our input referred mismatch model, which has its calculated equivalent variance defined in Eq. 3.25, adding at the output of ideal FFT, shown as a dashed line in all figure in this section. The mismatch of transistor pairs considered a normal random variable with zero mean and variance of δ_c^2 for strong inversion and a log-normal random variable for weak inversion based on the discussion in the previous chapter.

Normal Distributed Mismatch for Strong Inversion

In Fig. 4.8 we simulated an 8-FFT for different V_{th} variances. For instance the variation of $50(mV)$ is 10% of $V_{th} = 500(mV)$. For this variation we can find the variance $\delta_{c10\%}^2 = (0.1/3)^2 = 1.1 \cdot 10^{-3}$. Likewise for variations of 2.2% and 1%, the variance can be calculated as $5 \cdot 10^{-5}$ and $1.1 \cdot 10^{-5}$ respectively.

As it is clear in Fig. 4.8, The FFT is very sensitive to mismatch. On this plot as we increase the ΔV_{th} from 1% to 10%, the performance degrades severely. The performance loss due to 2.2% mismatch is about 3 dB at BER 10^{-3}, and for 10% mismatch the BER curve is almost flat.

In Fig. 4.9 we simulated 8-FFT up to 64-FFT for V_{th} variation of 1%. And in Fig. 4.10 we simulated 64-FFT up to 256-FFT for V_{th} variation of 0.1%. Referring

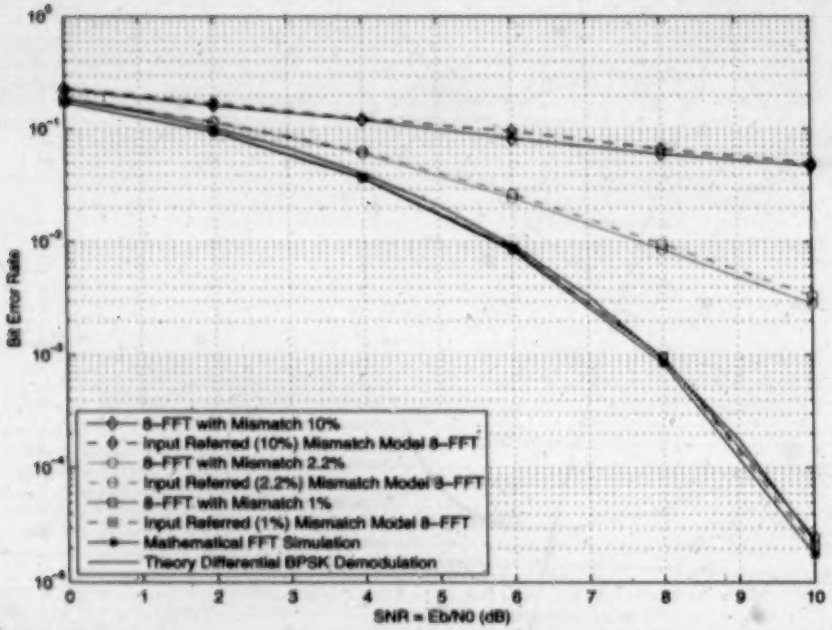
Bit Error Rate
SNR = Eb/N0 (dB)
8–FFT with Mismatch 10%
Input Referred (10%) Mismatch Model 8–FFT
8–FFT with Mismatch 2.2%
Input Referred (2.2%) Mismatch Model 8–FFT
8–FFT with Mismatch 1%
Input Referred (1%) Mismatch Model 8–FFT
Mathematical FFT Simulation
Theory Differential BPSK Demodulation

Figure 4.8: Mismatch simulation of 8-FFT for different threshold voltage variations, 10%, 2.2%, and 1%.

to these two simulations, we can observe that after four stages in the butterfly structure the impact of mismatch on FFT performance will cause an almost flat BER curve. This high degradation in performance depends on the number of previous stages in butterfly structure of FFT, explained on mismatch model section based on Eq. 3.25. Therefore as a next step it is worthwhile to change the butterfly structure to have a smaller number of stages. For example we can combine two successive stages in a butterfly structure having advantage of free addition by adding four currents at each node in each stage rather than two currents, which we used in existing design.

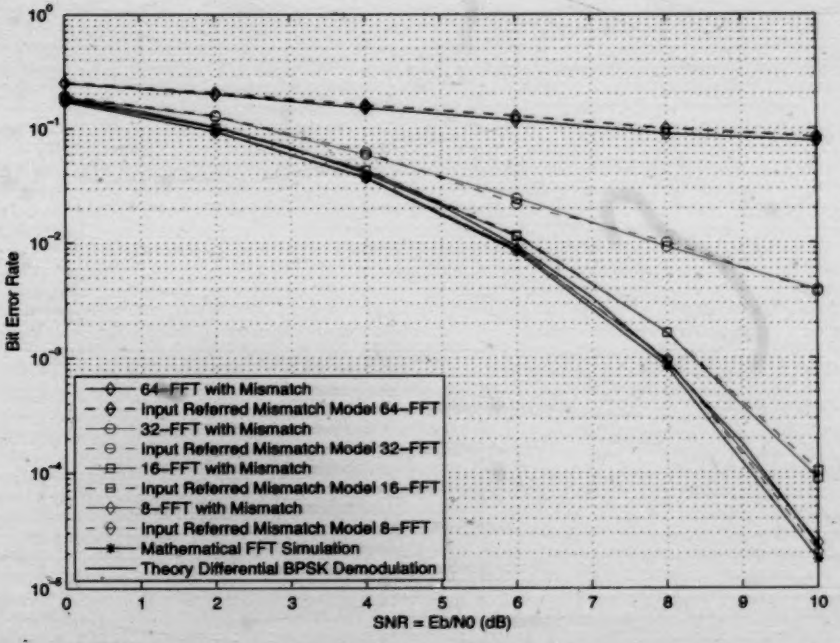

Figure 4.9: Mismatch simulation of FFT, from 8 to 64-FFT with 1% mismatch.

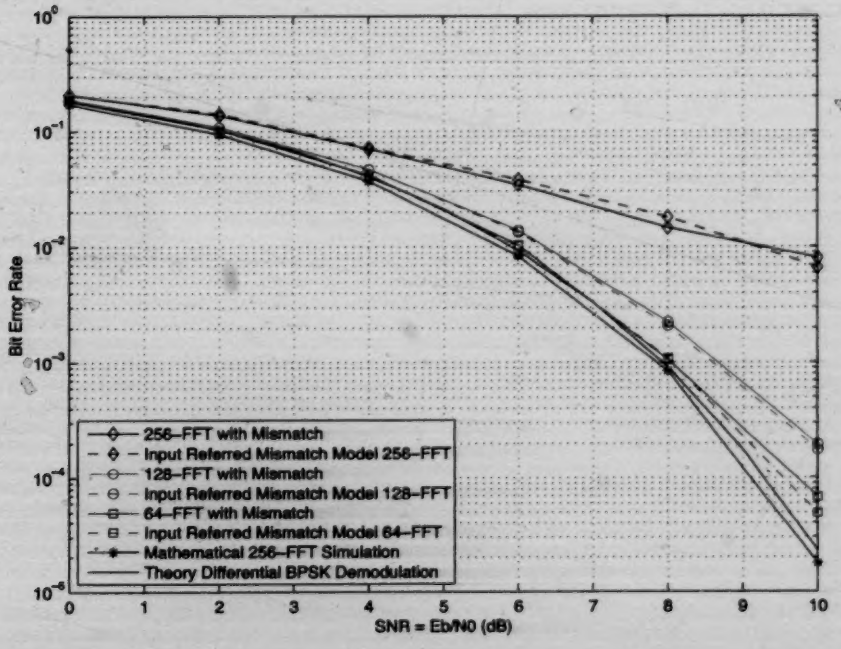

Figure 4.10: Mismatch simulation of FFT, from 64 to 256-FFT with 0.1% mismatch.

Log-normal Distributed Mismatch for Weak Inversion

Fig. 4.11 demonstrates the performance of an 8-FFT operating in weak inversion for different values of mismatch. If we compare the results shown in this figure to those of Fig. 4.8, we find that the sensitivity of FFT in weak inversion is much more than its sensitivity in strong inversion as we explain it in previous chapter.

In Fig. 4.8, for strong inversion the BER curve for 10 % mismatch is about 10^{-1} at SNR 10 dB, however for weak inversion in Fig. 4.11 the BER curve for 0.22 % mismatch is also about 10^{-1} at SNR 10 dB.

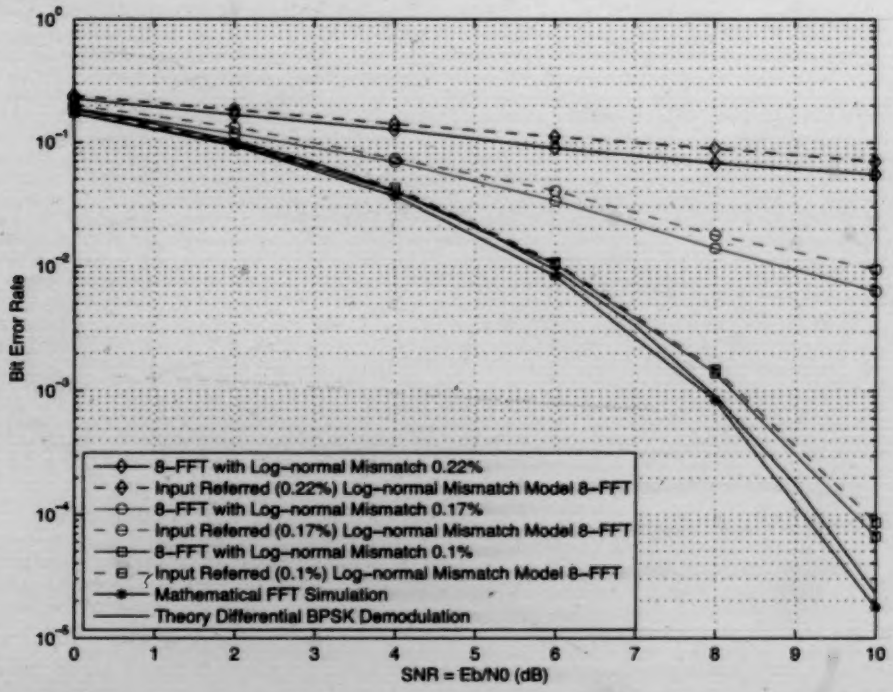
Bit Error Rate
SNR = Eb/N0 (dB)
8–FFT with Log–normal Mismatch 0.22%
Input Referred (0.22%) Log–normal Mismatch Model 8–FFT
8–FFT with Log–normal Mismatch 0.17%
Input Referred (0.17%) Log–normal Mismatch Model 8–FFT
8–FFT with Log–normal Mismatch 0.1%
Input Referred (0.1%) Log–normal Mismatch Model 8–FFT
Mathematical FFT Simulation
Theory Differential BPSK Demodulation

Figure 4.11: Mismatch simulation of 8-FFT in weak inversion for different threshold voltage variations, 0.22%, 0.17%, and 0.1%.

4.2.5 Radix-4 and Radix-16 Estimated Mismatch Model Simulations for 256-FFT

As we explained in previous chapter, to mitigate the impact of mismatch we can use higher radix structure to implement FFT with less number of stages since the

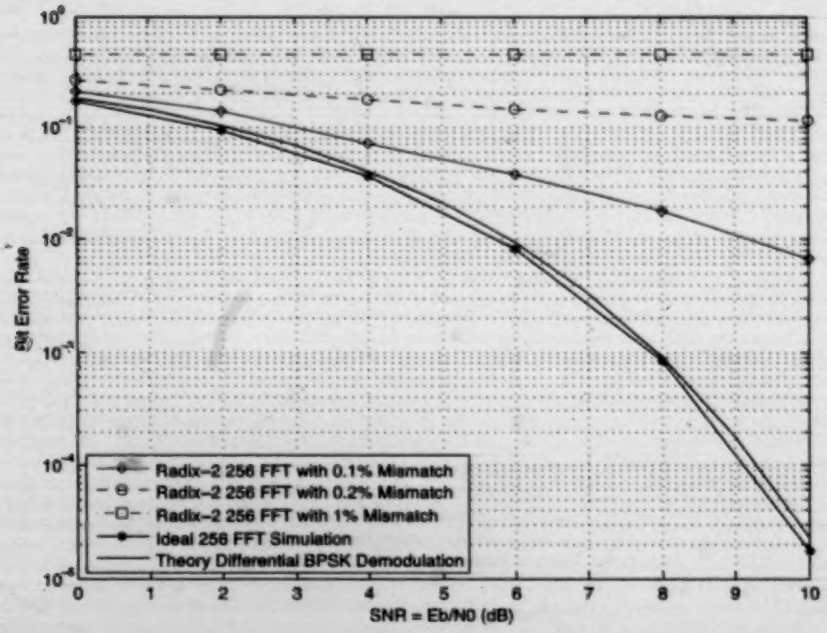
Bit Error Rate
SNR = Eb/N0 (dB)
Radix-2 256 FFT with 0.1% Mismatch
Radix-2 256 FFT with 0.2% Mismatch
Radix-2 256 FFT with 1% Mismatch
Ideal 256 FFT Simulation
Theory Differential BPSK Demodulation

Figure 4.12: Mismatch simulation of radix-2 256-FFT for 0.1%, 0.2%, and 1%, mismatch.

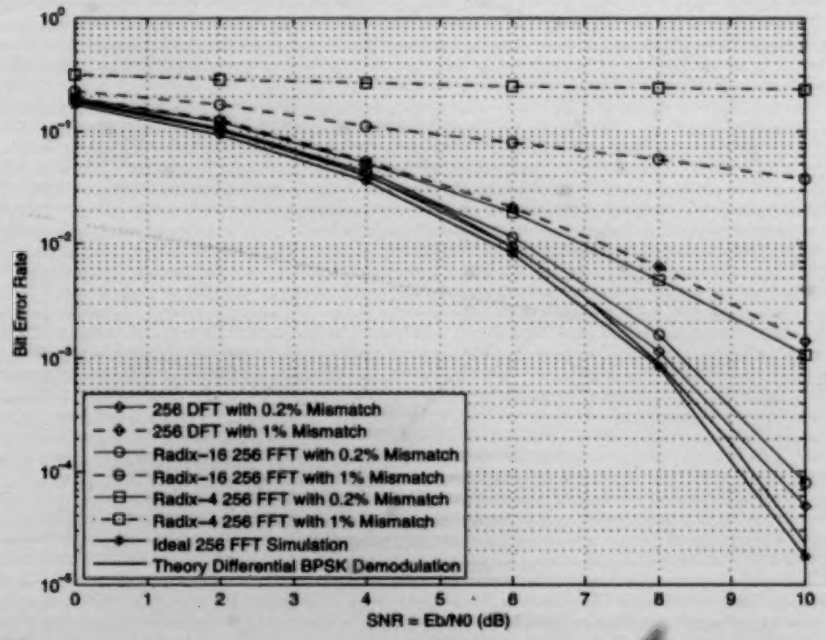
Bit Error Rate
SNR = Eb/N0 (dB)
256 DFT with 0.2% Mismatch
256 DFT with 1% Mismatch
Radix-16 256 FFT with 0.2% Mismatch
Radix-16 256 FFT with 1% Mismatch
Radix-4 256 FFT with 0.2% Mismatch
Radix-4 256 FFT with 1% Mismatch
Ideal 256 FFT Simulation
Theory Differential BPSK Demodulation

Figure 4.13: Mismatch simulation of 256-FFT for radix-4, radix-16 and DFT structure.

number of stages directly increases the impact of mismatch due to the discussion on our input referred mismatch model.

We simulated the model for radix-4, radix-16 and DFT that we have calculated previously, shown on Table 3.7. The input referred mismatch variance of DFT is roughly two orders of magnitude less than that of a radix-2 256-FFT, therefore DFT structure for 256-FFT with 10 times in mismatch percentage give us about the same performance of the radix-2 structure for the 256-FFT. For example, if we compare the plots shown in Fig. 4.12 and Fig. 4.13, we will see that the BER performance of a radix-2 256-FFT with 0.1% mismatch, as shown in Fig. 4.12, is about the same as the BER performance of a DFT with 1% mismatch, as shown in Fig. 4.13.

The difference between the performance of radix-4, radix-16 and DFT structures for the 256-FFT is shown in Fig. 4.13. As we can also observe from their input referred mismatch variance shown on Table 3.7, the radix-4 structure for the 256-FFT gives us about 4 times the input referred mismatch variance compared to the radix-16 structure and the radix-16 structure for the 256-FFT also gives us about 4 times the input referred mismatch variance compared to the DFT structure.

4.2.6 Decoder Simulations

We use a Hamming $(16, 11)^2$ Turbo Product Code (TPC), which has written by another PhD candidate [33], to simulate our 256-FFT and see how the decoder affects the performance of the 256-FFT for different values of mismatch. In Fig. 4.14 we simulated the radix-2 256-FFT having 0.1%, 0.22%, and 1% mismatch using TPC Decoder for BPSK Demodulation. Comparing the performance of 256-FFT with 0.1% mismatch without using decoder, shown in Fig. 4.12, with the corresponding performance having decoder shown here, we will find that decoder effectively mitigates the mismatch loss. For instance the 4 dB loss at BER of 10^{-2} for 1% mismatch turns to half a dB loss at BER of 10^{-2} up to 10^{-4}.

For higher percentage of mismatch the decoder does not have that much improving impact. To obtain a better performance one idea is to give the output of our input referred mismatch block to the decoder as an external source of noise. By doing this decoder includes this noise as a model of the new channel for finding the

probability of total noise in each iteration.

If we use a DFT we obtain roughly the same performance shown but for 1%, 2.2%, and 1% mismatch respectively.

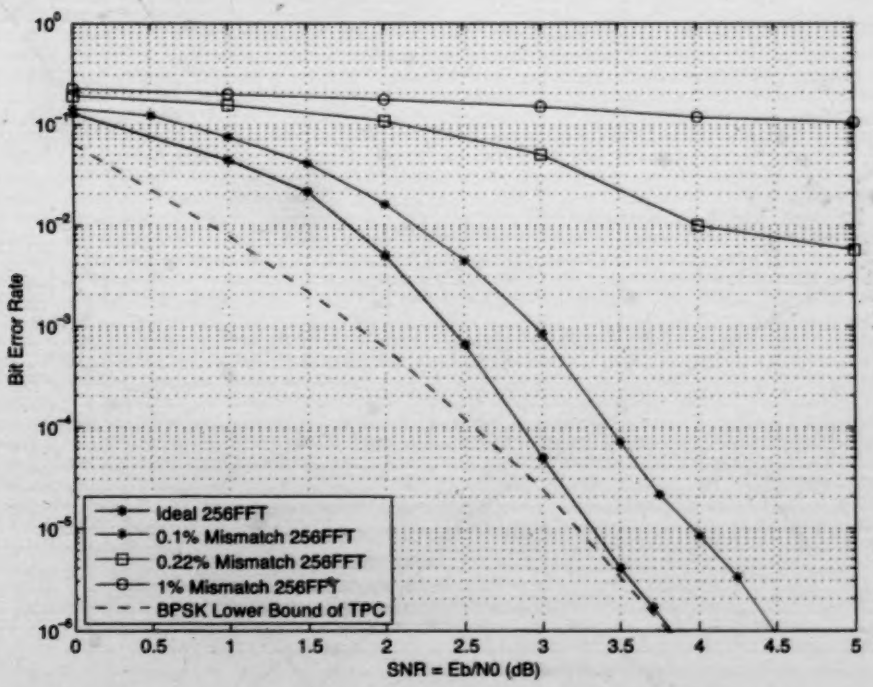

Figure 4.14: Radix-2 256-FFT performance using TPC Decoder for BPSK dimodulation

4.3 System Performance Matrix

We provide the system performance matrix shown in Table 4.1 to present design trade offs on the power consumption and the silicon area for different mismatch values and FFT types having a desirable BER performance. The idea is to increase the V_{gs} of transistors to decrease the normalizing factor in strong inversion, Con_{str}, which is equal to $\frac{2V_{th}}{V_{gs}-V_{th}}$. Increasing the V_{gs} will cost an extra power consumption since we have to increase the V_{dd} as well. Also by increasing the V_{gs} we need to increase the length of transistors, L, to maintain our desirable bias currents; hence extra silicon area is required as well.

256-FFT Type	Mismatch	Con_{str}	F_{imp}	$V_{dd}(V)$	P_{factor}	S_{factor}
Radix-2	10%	-	1	-	-	-
Radix-4	10%	-	3	-	-	-
Radix-16	10%	0.06	6	27.5	15.28	278
DFT	10%	0.12	12.5	14.1	7.83	69
Radix-2	5%	-	1	-	-	-
Radix-4	5%	0.06	3	27.5	15.28	278
Radix-16	5%	0.12	6	14.1	7.83	69
DFT	5%	0.25	12.5	7.2	4.00	16
Radix-2	1%	0.1	1	16.8	9.33	100
Radix-4	1%	0.3	3	6.1	3.39	11
Radix-16	1%	0.6	6	3.4	1.89	2.6
DFT	1%	1.25	12.5	2.1	1.17	0.7

Table 4.1: The system performance matrix for having BER loss of 0.5 dB for different values of V_{th} mismatch. Con_{str} is $\frac{2V_{th}}{V_{gs}-V_{th}}$, F_{imp} is the standard deviation improving factor for different FFT types, P_{factor} is a factor representing an extra power consumption due to increasing V_{dd} from 1.8 (V), and S_{factor} is a factor representing an extra Silicon area due to increasing L from minimum feature size, $0.18\mu m$.

In Table 4.1 we assumed the 0.5 dB loss for the TPC decoder BER performance. To satisfy this performance we find the Con_{str} by comparing the input mismatch variances calculated in Table 3.7. Having a specific value of Con_{str}, we calculate the V_{gs}. Then we calculate the V_{dd} and L, the length of transistors, to find the cost of power and area for having this performance for different FFT types. The P_{factor}, $\frac{V_{dd}}{1.8}$, is a factor representing an extra power consumption due to increasing V_{dd} from 1.8 (V) for constant input bias currents, and the S_{factor} is a factor representing an extra silicon area due to increasing L from minimum feature size, $0.18\mu m$ to have such constant input bias currents.

For example, if we consider the 256-FFT type of radix-2 with 1% mismatch in Table 4.1, to have a BER performance loss about 0.5 dB we want to find a required Con_{str}. To do so, we look at the BER curve shown in Fig. 4.14, it is 0.5 dB off but this curve is for 0.1% mismatch. Based on Eq. 3.16 we need a Con_{str} equal to 0.1 to compensate this factor 10. It leads to $V_{gs} = 21 \cdot V_{th}$. If we assume $V_{th} = 400$ (mV), V_{gs} turns to be 8.4 (V). Since we have two transistors on top off each-other at every V_{dd} to $V_{ss} = 0$ path, shown in Fig. 3.12, and we like them to operate in the

saturation region, the V_{dd} should be $2 \cdot V_{gs} = 16.8\ (V)$. To find S_{factor}, we know that L is proportional to $\frac{(V_{gs}-V_{th})^2}{(V_{gs-nom}-V_{th})^2}$, where V_{gs-nom}, the assumed value for V_{gs} for minimum feature size, is equal to $3 \cdot V_{th}$.

For other types of FFT illustrated in Table 4.1 we follow the same steps to find all different given parameters. The only thing we need to take into account is that to obtain these parameters for other types of FFT from the BER performance shown in Fig. 4.14, we need to find the proper coefficient due to their different input referred mismatch variances given in Table 3.7. For instance, if we consider the DFT with 1% mismatch instead of the radix-2 256-FFT at which we just looked, we also need to calculate $\sqrt{\frac{251 \cdot (10^3) \cdot \delta_x^2}{1.61 \cdot (10^3) \cdot \delta_x^2}} = 12.5$ as an improving factor, F_{imp}, for the standard deviation of the ε, which can be added in Eq. 3.16 to obtain $(1 + Con_{str} \cdot \frac{1}{F_{imp}} \cdot \varepsilon)$.

4.4 Chapter Conclusion

In this Chapter, we showed that for analog front end processing weak inversion operation is too sensitive to the mismatch of transistor pairs, so we do not want to operate in this region. To build an analog FFT we showed that the higher radix FFT structures outperform compared to the radix-2 FFT. The simplification of WFs is not beneficial considering the tradeoff between BER performance gain and design complexity. The output of our mathematical input referred mismatch model for the FFT matches the simulated FFT output, which has mismatch at each current mirror. Hence it is a useful tool to analyze the analog FFT processor.

Chapter 5

Conclusions

The idea of designing analog decoders recently arose out of work on capacity approaching codes such as Turbo codes and LDPC codes. Therefore the idea of an entire analog receiver including the decoder and all other interfaces on a single chip has been studied to replace the traditional digital design. Although digital processors are very fast and accurate, they are very power hungry and need a large amount of silicon. The parallel nature of analog design in iterative algorithms makes it more suitable for decoder designs than digital circuits, without excessive power consumption of the digital circuits due to high frequency switching and with acceptable speed due to parallel structures. The fundamental goal of our project is to move towards system-level integration of analog decoders with other basic communications receiver components, while maintaining the power consumption advantages of analog decoders to make them suitable for use with energy scavenging methods.

5.1 Contributions

We designed and simulated an analog 256-FFT as an input interface to the analog decoder in an analog receiver. Our design has considered both system and circuit issues. At the system level we chose an OFDM communication transceiver model using differential BPSK modulation. At the circuit level we modeled the basic blocks of analog FFT, current mirrors, and explained how we can use them to build an analog FFT. We considered different circuit issues like choosing bias currents

to define the operational region of transistors, strong or weak inversion, and we proposed the idea to have the same input and output current range to save on power consumption, and we examined the performance of a simplified FFT by having different values for WFs to simplify the circuit.

Considering the mismatch of each transistor pair inside the analog FFT as a Gaussian random variable, we modeled an input referred mismatch source of noise for an entire FFT which is also a Gaussian random variable with a calculated variance from the mismatch variance. This model is very simple and fairly accurate based on the simulation results.

We simulated our analog 256-FFT using our proposed communication system. We analyzed the BER performance of the system considering different circuit issues like different bias currents, having current scaling at each FFT stage to obtain the same input/output current range, simplification of weighting factors inside the FFT structure, and different mismatch values for each transistor pair inside the analog FFT.

We examined our input referred mismatch model using two different simulations: first we used our analog 256-FFT which has mismatch at each transistor pair, and in the second simulation we used an analog FFT without mismatch and we added the output of our input referred mismatch model to the output of the analog FFT; both BER simulations matched.

We also examined our analog FFT using a TPC decoder to find how the decoder mitigates the effect of mismatch on BER performance for our analog FFT. The decoder decreased the performance loss due to the circuit mismatch. Therefore the analog 256-FFT could be used as an input interface for our analog decoder to satisfy the low power consumption constraint.

Simulation Conclusions

At the current mirror model and current scaling simulation sections for different bias currents, we discussed how we can compensate the offset due to the specific bias current by slightly changing the (W/L) of the output transistors of each pair. This provides us a better BER performance, however it means that the 256-FFT is

very sensitive to minor changes of the physical parameters of transistors.

The simplification of WFs in the analog FFT is not remarkably beneficial, as discussed before, compared to its cost of BER performance loss, so we try to create fairly accurate WFs to minimize this loss.

Mismatch simulations shows that the analog FFT operating in weak inversion has a mismatch variance about two orders of magnitude larger than that of analog FFT working in strong inversion. Therefore analog FFT operating in weak inversion is more sensitive to mismatch compared to that of strong inversion operation.

The radix-2 analog 256-FFT, the proposed FFT butterfly structure in this research, is highly sensitive to the mismatch of transistor pairs, due to the high number of stages in its butterfly structure, 8 stages. We calculated the input referred mismatch model for higher radix FFT structures. We showed that for a DFT implementation of the 256-FFT, which has only 1 stage, we achieve about two order of magnitude less equivalent variance of our model compared to that of the radix-2 analog 256-FFT. Also the number of current mirrors in DFT is less than that of the other different radix 256-FFT structures. Hence the DFT in analog design is the best option compared to the other FFT structures.

The TPC decoder has an effective improvement on the BER performance of the analog 256-FFT. The BER performance of a radix-2 analog 256-FFT with mismatch of 0.1%, which is $\pm 0.5(mV)$ tolerance in $V_{th} = 500(mV)$, is only half a dB off compared to an analog 256-FFT without mismatch. If we use a radix-16 FFT structure instead, we obtain the same performance but for mismatch of 1%, which is $\pm 5(mV)$ tolerance in $V_{th} = 500(mV)$.

5.2 Future Work

There are several potential research directions to continue this thesis. The main directions of the future work include:

1. Input Referred Mismatch Model as Extra Noise for Decoder Use

To estimate the final value of the received bits, the decoder uses the channel noise PDF and performs several iterations to obtain the final probability of being 1 or

-1 based on the LLR values. By considering our input referred mismatch model as extra channel noise and estimating the PDF of this additive noise at the output of the ideal FFT, we can modify the channel noise, AWGN, PDF used inside the decoder to provide the decoder a better estimation of our new channel noise to improve the BER performance.

2. Higher Radix FFT Design

Based on the outcome of this thesis, the radix-2 FFT butterfly structure is highly sensitive to the mismatch of transistor pairs due to the large number of sub-stages used in this structure. Higher radix FFTs, radix-4 , radix-16 and DFT, show a better BER performance for different mismatch variances. A new circuit implementation of an analog FFT based on a higher radix FFT structure will be the next step.

3. Optimum Bias Current for Noise versus Speed Analysis

To operate at high frequency, we need to increase the bias currents. The equivalent noise current however gets increased in higher frequencies. Also at a very low amount of bias currents (p-Amp) circuit provides more noise. Therefore finding the optimum bias currents to minimize the equivalent noise current versus frequency is another research direction for this thesis.

4. Fabrication and Test

We have already fabricated a test chip of the analog FFT using 180-nm CMOS technology in collaboration with another M.Sc. candidate. Testing of this chip is the next step.

The intention is to design, fabricate, and test a complete baseband analog receiver in CMOS technology to verify the design.

Bibliography

[1] C. Winstead. *Analog Iterative Error Control Decoders*. PhD thesis, University of Alberta, Edmonton, Alberta, Canada, Pages 4-8, 111, 2004.

[2] N. Sadeghi, H. M. Nik, C. Schlegel, V. C. Gaudet, and K. Iniewski. *Analog FFT Interface for Ultra-Low Power Analog Receiver Architectures*. Analog Decoding Workshop, Pages 11-14, June 2006.

[3] C. Berrou, A. Glavieux, and P. Thitimajshima. *Near Shannon Limit Error Correcting Coding and Decoding: Turbo Codes*. IEEE Transaction on Information Theory, Number 44, Pages 1261-1271, October 1996.

[4] R. G. Gallager. *Low-Density Parity-Check Code*. IRE Transaction on Information Theory, Volume IT-8, Pages 21-28, January 1962.

[5] T. Ohtsuki. *LDPC Codes in Communications and Broadcasting*. IEEE Transactions on Communications, Volume 90-B, Number 3, Pages 440-453, March 2007.

[6] A. J. Blanksby and C. J. Howland. *A 690-mW 1-Gb/s 1024-b, Rate-1/2 Low-Density Parity-Check Code Decoder*. IEEE Journal of Solid-State Circuits, Volume 37, Number 3, Pages 404-412, March 2002.

[7] C. Winstead, N. Nguyen, V. C. Gaudet, and C. Schlegel. *Low-Voltage CMOS Circuits for Analog Iterative Decoders*. IEEE Transactions on Circuits and Systems, Volume 53, Number 4, Pages 829-841, April 2006.

[8] S. Hemati, A. H. Banihashemi, and C. Plett. *A 0.18-um CMOS Analog Min-Sum Iterative Decoder for a (32,8) Low-Density Parity-Check (LDPC) Code*.

IEEE Journal of Solid-State Circuits, Volume 41, Number 11, Pages 2531-2540, November 2006.

[9] C. Winstead, J. Dai, S. Yu, C. Myers, R. R. Harrison, and C. Schlegel. *CMOS Analog MAP Decoder for (8,4) Hamming Code*. IEEE Journal of Solid-State Circuits, Volume 39, Number 1, Pages 122-131, January 2004.

[10] J. A. Paradiso and T. Starner. *Energy Scavenging for Mobile and Wireless Electronics*. IEEE Communication Society, http://www.media.mit.edu, 2005.

[11] D. Vogrig, A. Gerosa, A. Neviani, A. G. Amat, G. Montorsi, and S. Benedetto. *A 0.35- m CMOS Analog Turbo Decoder for the 40-bit Rate 1/3 UMTS Channel Code*. IEEE Journal of Solid-State Circuits, Volume 40, Number 3, Pages 753-762, March 2005.

[12] V. Gaudet and G. Gulak. *A 13.3 Mbps 0.35um CMOS Analog Turbo Decoder IC with a Configurable Interleaver*. IEEE Journal of Solid-State Circuits, Volume 38, Number 11, Pages 2010-2015, November 2003.

[13] M. M. Mansour and N. R. Shanbhag. *A 640-Mb/s 2048-bit Programmable LDPC Decoder Chip*. IEEE Journal of Solid-State Circuits, Volume 41, Number 3, Pages 684-498, March 2006.

[14] C. E. Shannon. *A mathematical theory of communication*. Bell Syst. Tech. J., Volume 27, Pages 379-423, 623-656 1948.

[15] J. Hagenauer, E. Offer, and L. Papke. *Iterative Decoding of Binary Block and Convolutional Codes*. IEEE Transaction on Information Theory, Volume 42, Number 2, Pages 429-445, March 1996.

[16] R. Pyndiah. *Near Optimum Decoding of Product Codes: Block Turbo Codes*. IEEE Transaction on Information Theory, Volume 42, Number 8, Pages 1003-1010, August 1998.

[17] D. J. C. Mackay. *Good Error-Correcting Codes Based on Very Sparse Matrices*. IEEE Transactions on Information Theory, Volume 45, Pages 399-431, March 1999.

[18] T. J. Richardson, M. Shokrollahi, and R. Urbanke. *Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes*. IEEE Transactions on Information Theory, Pages 619-637, February 2001.

[19] N. Sadeghi, S. Howard, S. Kasnavi, K. Iniewski, V. C.Gaudet, and C. Schlegel. *Analysis of Error Control Code use in Ultra-Low-Power Wireless Sensor Networks*. IEEE International Symposium on Circuits and Systems Conference, Pages 3558-3561, May 2006.

[20] R. V. Nee and R. Prasad. *OFDM for Wireless Multimedia Communications*. Artech House, Boston, 2000.

[21] D. Z. Liu and C. H. Wei. *DAPSK-OFDM transmissions for high date-rate digital mobile radio*. IEEE International Symposium on Circuits and Systems, Volume 2, Pages 417-420, May 2001.

[22] J.G. Proakis. *Digital Communications*. McGraw-Hill, New York, fourth edition, Pages 254-257, 272-276, 2001.

[23] S. C. Liu, Kramer, Indiveri, Delbruck, and Douglas. *Analog VLSI: Circuits and Principles*. The MIT Press, Cambridge, Massachusetts, 2002.

[24] M. Hasan and T. Arslan. *Implementation of Low-Power FFT Processor Cores Using a Novel Order-Based Processing Scheme*. IEE Proceedings on Circuits, Devices and Systems, Volume 150, Number 3, Pages 149-154, June 2003.

[25] R. S. Sherratt, O. Cadenas, and N. Goswami. *A Low Clock Frequency FFT Core Implementation for Multiband Full-Rate Ultra-Wideband (UWB) Receivers*. IEEE Transactions on Consumer Electronics, Volume 51, Number 3, Pages 798-802, August 2005.

[26] W.C. Yeh and C.-W. Jen. *High-Speed and Low-Power Split-Radix FFT*. IEEE Transactions on Signal Processing, Volume 51, Number 3, Pages 864-874, March 2003.

[27] A. V. Oppenheim, R. W. Schafer, and J. R. Buck. *Discrete-Time Signal Processing*. Prentice-Hall Inc., New Jersey, second edition, Pages 629-669, 1999.

[28] P. Rosel. *Timing of Some Bit Reversal Algorithms*. Elsevier North-Holland, Inc., Volume 18, Pages 425-433, 1989.

[29] M. J. M. Pelgrom, A. C. J. Duinmaijer, and A. P. G. Welbers. *Matching Properties of MOS Transistors*. IEEE Journal of Solid-State Circuits, Volume 24, Number 5, Pages 1433-1440, October 1989.

[30] P. G. Drennan and C. C. McAndrew. *Understanding MOSFET Mismatch for Analog Design*. IEEE Journal of Solid-State Circuits, Volume 38, Number 3, Pages 450-456, March 2003.

[31] A. Oruganti and N. Ranganathan. *Leakage Power Reduction in Dual-Vdd and Dual-Vth Designs through Probabilistic Analysis of Vth Variation*. IEEE International Conference on VLSI Design, Pages 1-4, 2006.

[32] Y. Tsividis. *Operation and Modeling of The MOS Transistor*. Oxford University Press, New York, second edition, Pages 170-175, 1999.

[33] H. Moussavinik. *TPC Iterative Decoding with Flooding and Row-Column Updating Schedule*. Master's thesis, Chalmers University of Technology, Pages 29-31, 2006.